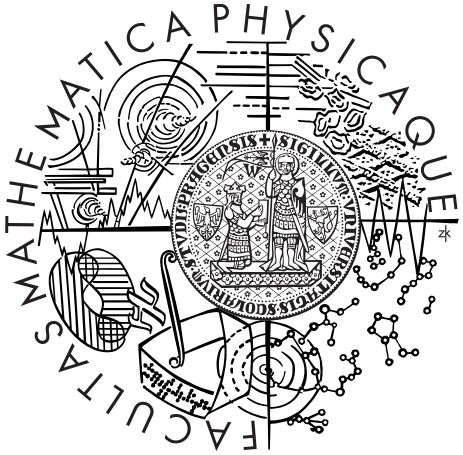
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Miroslav Štola

Risk

Katedra aplikované matematiky

Vedoucí bakalářské práce: RNDr. Josef Cibulka, Ph.D.
Studijní program: Informatika
Studijní obor: Obecná informatika (IOI)

Praha 2013

Děkuji RNDr. Josefu Cibulkovi, Ph.D. za trpělivé vedení práce, za cenné rady a podněty. Dále bych rád poděkoval Lucii Baťkové za asistenci s grafikou. Díky patří také všem, kteří hru ochotně testovali.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 20.5.2013 Podpis autora

Název práce: Risk

Autor: Miroslav Štola

Katedra: Katedra aplikované matematiky

Vedoucí bakalářské práce: RNDr. Josef Cibulka Ph.D., Katedra aplikované matematiky

Abstrakt: Tato práce si klade za cíl implementovat strategickou deskovou hru Risk s pozměněnými pravidly a vytvořit k ní umělou inteligenci. Hru je možné hrát v režimu hot-seat na jednom počítači v případě většího počtu lidských hráčů. Umělá inteligence je řešena pomocí prohledávání stavového stromu. Dále byla vyvinuta přímočará umělá inteligence. Obě umělé inteligence byly testovány proti sobě i proti lidem. Na konci práce jsou uvedeny a diskutovány výsledky.

Klíčová slova: umělá inteligence, tahová desková hra, prohledávání

Title: Risk

Author: Miroslav Štola

Department: Department of Applied Mathematics

Supervisor: RNDr. Josef Cibulka, Ph.D., Department of Applied Mathematics

Abstract: The main goal of this thesis is to implement a strategy board game called Risk with modified rules and to create an artificial intelligence. It is possible to play the game in the hot-seat mode on one computer in case of multiple human players. The artificial intelligence is based on a search tree. Additionally, a straightforward artificial intelligence was created. Both AI players are tested against one another and also against human players. At the end of the thesis the results are presented and discussed.

Keywords: artificial intelligence, turn-based board game, search

Obsah

Úvod

0.1 Letmý pohled do historie

Počátky původní deskové hry Risk sahají do roku 1957[2], kdy ve Francii vyšla pod názvem *La Conquête du Monde* (Dobytí světa). Hru navrhl scenárista a režisér v jedné osobě Albert Lamorisse, známý například pro svůj počin *The Red Balloon*.

Ke konci padesátých let vydává společnost Parker Brothers verzi pro Spojené státy americké. Postupem času vznikaly další verze inspirované dějinami, vizí budoucnosti či fantasy fikcí. Jmenovitě například: Napoleonská edice, Hvězdné války, Pán prstenů, Risk 2022 AD, kde lze dobýt i Měsíc. Více informací je dostupných na internetové stránce věnované historii hry Risk [2]. V průběhu času vzniklo i mnoho neoficiálních distribucí a map. Oficiálním vlastníkem práv ke hře je v současné době společnost Hasbro[3]. Velké obohacení přinesla verze z roku 1993, která představila systém tajných misí, jejž převzala i tato práce, více viz další odstavec.

0.2 Základní informace o hře

Risk je stolní desková hra, která stylizuje hráče do role vojevůdců. Lze ji hrát ve 2-5 hráčích. Každý hráč začíná s jedním územím, odkud se snaží expandovat a dobývat další země. Cílem hry je buď dobýt celý svět, nebo splnit tajný úkol, který si každý hráč na začátku hry vylosuje. Hra prověří strategické schopnosti hráče, svou roli však hraje i náhoda, neboť nedílnou součástí hry je také snímání karet a hody kostkami. Tato práce pojednává o pozměněné verzi, která kupříkladu kompletně mění systém revolucí. Přidává také neutrální hráče a hlavní města. Podrobně si rozebereme pravidla v první kapitole.

Terminologická poznámka Nadále se v práci budou vyskytovat termíny „země“, „teritorium“, „provincie“, „stát“. Všechny mají stejný význam jako „území“.

0.3 Struktura práce

Tato práce se zabývá převedením pozměněné hry Risk do hratelné elektronické podoby pomocí programovacího jazyka Java. Počítačová verze umožňuje hrát v režimu hot-seat na jednom stroji, případně proti jednomu nebo více umělým hráčům dvou typů. Práce je členěna do čtyř hlavních kapitol vyjma úvodu.

Pravidla hry V první kapitole jsou vysvětleny nejen odlišnosti od původní stolní verze, ale i kompletní pravidla platná pro změněnou verzi, přehledně členěna podle fází tahu.

Uživatelské rozhraní a ovládání Ve druhé kapitole je poskytnut návod, jak aplikaci zprovoznit na počítači. Dále se věnuje ovládání programu a grafickému uživatelskému rozhraní.

Implementace Třetí kapitola je zasvěcena implementaci programu samotného. Je v ní rozebrán návrh herního jádra, důležitých tříd a popis řízení AI tahu v grafickém rozhraní.

Umělá inteligence Ve čtvrté kapitole se do hloubky seznámíme s umělou inteligencí pro počítačem řízeného hráče. Popisuje evaluační funkci prohledávacího stromu, způsob generování možných tahů a příklady situací, ve kterých si AI nepočíná rozumně. Také se soustředí na implementační stránku umělé inteligence. Rovněž se věnuje problémům, které bylo třeba překonat, aby prohledávání mohlo správně fungovat a vracelo tahy v rozumném čase. Stručně je popsána umělá inteligence přímočarého hráče. Dále uvedeme experimentální výsledky, jak si počítačem řízení hráči vedli proti lidem i proti sobě navzájem.

Závěr V závěrečné kapitole je celý projekt zhodnocen. Dále jsou navržena nerealizovaná vylepšení a případné plány do budoucna. Zmíněny jsou podobné implementace.

1. Pravidla

Tato kapitola pojednává o pravidlech změněné verze hry. Pro úplnost jsou na konci kapitoly uvedeny hlavní odlišnosti od klasické deskové předlohy.

1.1 Součásti hry

Originální balení stolní hry obsahuje zejména mapu světa, 5 hracích kostek, plastové vojáky, balíček karet revoluce a karty s tajným úkolem.

Mapa Svět se v Risku člení na kontinenty a ty se skládají z jednotlivých území. Za každé území, které hráč vlastní, získává více jednotek v každém dalším tahu. Pokud ovládá celý kontinent, získá větší bonus jednotek za kolo. Více v následující tabulce.

Jméno kontinentu	**Seznam území**	**Bonus**
North America	Alaska, Northwestern Territory, Greenland, Alberta, Ontario, Quebec, Western US, Eastern US, Central Amerika	5
Europe	Iceland, Scandinavia, Ukraine, Great Britain, Northern Europe, Western Europe, Southern Europe	5
Asia	Ural, Siberia, Yakutsk, Kamchatka, Irkutsk, Mongolia, Japan, Afghanistan, China, Middle East, India, Siam	7
South America	Venezuela, Brazil, Peru, Argentina	2
Africa	Northern Afrika, Egypt, Eastern Africa, Congo, South Afrika, Madagascar	3
Australia	Indonesia, New Guinea, Western Australia, Eastern Australia	2

Tabulka 1.1: Kontinenty

Kostky Pomocí kostek se určuje vítěz bitev. Útočník disponuje až třemi kostkami, obránce dvěma. Podrobněji se s dobýváním seznámíme v sekci 1.4.3.

Karty revoluce Pro každé území obsahuje hra kartu s názvem teritoria a vyobrazením pěšáka, jezdce nebo dělostřelectva. Obrázek určuje sílu karty. Obrázky na kartách se však vyskytují pouze v původní stolní verzi, tento projekt si vystačí s číselným vyjádřením síly. Karty se používají buď k posílení vlastního území, nebo k vyvolání revoluce v nepřátelské zemi. Obsáhleji se jim budeme věnovat dále v sekci 1.4.2.

Karty úkolů Každý hráč má tajný úkol, jehož splnění je jedním ze způsobů, jak hru vyhrát. Přehled možných úkolů je uveden níže.

Anglicky	**Význam**
Conquer NORTH AMERICA and AFRICA	Dobýt Severní Ameriku a Afriku
Conquer any 18 TERRITORIES, each must contain at least 2 units	Obsadit libovolných 18 území, v každém z nich mít minimálně dva vojáky
Conquer EUROPE, AUSTRALIA and a third continent to your liking	Dobýt Evropu, Austrálii a třetí, libovolný kontinent
Conquer ASIA and AFRICA	Dobýt Asii a Afriku
Conquer EUROPE, SOUTH AMERICA and a third continent to your liking	Dobýt Evropu, Jižní Ameriku a třetí, libovolný kontinent
Conquer ASIA and SOUTH AMERICA	Dobýt Asii a Jižní Ameriku
Conquer NORTH AMERICA and AUSTRALIA	Dobýt Severní Ameriku a Austrálii
Conquer 24 TERRITORIES to your liking	Ovládnout 24 libovolných teritorií
Eliminate player X	Eliminovat hráče X

Tabulka 1.2: Tajné úkoly

Za eliminovaného hráče se považuje takový hráč, který již neovládá žádné území. Hráče zničí ten hráč, jenž dobude poslední území daného hráče.

1.2 Před hrou

Než začne hra samotná, hráči si určí pořadí, v jakém budou hrát. Buď se dohodnou, nebo si všichni hodí kostkou a podle výsledku se seřadí. Každý hráč si vylosuje kartu s tajnou misí (viz Tabulka 1.2). Kartu si přečte a neukazuje ji žádnému z ostatních hráčů. Dále si každý z hráčů popořadě sejme ze zamíchaného balíčku karet revolucí tři karty. Vybere si jednu z nich a území odpovídající kartě prohlásí za své hlavní město, jež osadí sedmi jednotkami. Danou kartu si nechá a ostatní dvě vrátí do balíčku. V programu je přirozeně pořadí hráčů určeno náhodně. Totéž platí i pro tajný úkol a možnosti výběru hlavního města. Jakmile si vyberou všichni hráči své hlavní město, hra začne. Všechna území, která nebyla vybrána, se osídlí neutrálními jednotkami. Tyto jednotky samy o sobě nemohou útočit, ale napadené území budou bránit do posledního muže. Následující tabulka ukazuje, kolika jednotkami se jednotlivá území obsadí. Počet jednotek odpovídá i síle revolučních karet.

Název území	**Jednotek**
Alaska	3
Northwestern territory	7
Greenland	5
Alberta	3
Ontario	5
Quebec	7
Western US	3
Eastern US	7
Central America	5
Iceland	3
Scandinavia	7
Ukraine	7
Great Britain	5
Northern Europe	5
Western Europe	3
Southern Europe	5
Ural	5
Siberia	7
Yakutsk	5
Kamchatka	5
Irkutsk	3
Mongolia	7
Japan	3
Afghanistan	3
China	5
Middle East	7
India	3
Siam	7
Venezuela	7
Peru	5
Brazil	7
Argentina	3
Northern Africa	3
Egypt	3
East Africa	7
Congo	5
South Africa	7
Madagascar	3
Indonesia	5
New Guinea	5
Western Australia	7
Eastern Australia	3

Tabulka 1.3: Jednotky u karet revoluce

1.3 Průběh hry

- Hráči se po tazích střídají, dokud někdo nevyhraje.
- Pokud dojde balíček karet revoluce, zamíchá se odhazovací balíček a hráči si berou karty z něj. V případě, že nejsou k dispozici ani odhozené karty, hráč ztrácí nárok na nabytí karty.
- V situaci, kdy je hráč zničen jinak než dobytím hlavního města, přecházejí jeho karty do odhazovacího balíčku.

1.3.1 Pravidla týkající se hlavního města

Hlavní město u každé říše hraje důležitou roli. Pokud ho hráč ztratí, vítěz přebírá veškeré jeho karty revoluce a rovněž zabere všechny hráčovy země ležící na stejném kontinentu jako hlavní město. Tyto země obsadí jednou jednotkou. Z toho je zřejmé, že je potřeba své hlavní město pečlivě střežit.

1.3.2 Cíl hry

Hru vyhraje ten hráč, který ovládne celý svět, zničí všechny ostatní hráče nebo splní svou tajnou misi. Někdy se v průběhu hry může stát, že misi již nelze splnit. Například hráč X má za úkol zničit hráče Y, ten však již byl zničen hráčem Z. V tom případě dostane hráč X nový úkol. Pokud někdo již splnil svůj úkol, hra končí, ostatní nedostávají šanci dohrát poslední kolo.

1.4 Tah

Každý tah má několik fází: posilovací, revoluční, útočnou, pohybovou. Jakmile dokončí tah jeden hráč, táhne hráč následující.

1.4.1 Fáze posilování (reinforce)

V této fázi si může hráč zvýšit počet vojáků ve svých zemích. Počet jednotek, které mu budou k dispozici, se odvíjí od následujícího vztahu:

$$jednotek = \max\{3, počet\_území + bonus\_kontinent\}$$

Jinými slovy, dostane tolik jednotek, kolik ovládá území (minimálně však 3), a navíc další jednotky, pokud ovládá nějaký kontinent, viz Tabulka 1.1. Tyto jednotky mohou být poslány do libovolného vlastněného území a každý voják může být poslán jinam. Po přiřazení jednotek následuje fáze revoluce.

1.4.2 Fáze revoluce (revolution)

Nyní může hráč použít své revoluční karty. Avšak pouze ty, které získal v některém z minulých kol. Každá karta má určitou sílu: 3, 5 či 7 jednotek. Výčet karet naleznete v tabulce 1.3. Existuje jediná výjimka: pokud hrajeme kartu svého hlavního města, síla karty je vždy 7. Po použití putuje karta na odhazovací balíček.

Revoluční posilování Pokud karta reprezentuje zemi, kterou hráč vlastní, přibude mu v provincii stejný počet jednotek jako je síla karty, viz Tabulka 1.3. Jak bylo uvedeno výše, pokud se jedná o vlastní hlavní město, pak je síla vždy 7.

Revoluční útok V případě, že území odpovídající hrané kartě je okupováno cizím hráčem (či neutrálními jednotkami), dojde k takzvané revoluci. Útočící hráč (dále útočník) má k dispozici tolik jednotek, kolik je vyobrazeno na kartě. Bránící hráč (dále obránce) disponuje tolika jednotkami, kolik jich má v napadeném území.

Počet kostek Nyní se určí počet kostek, kterými mohou oba hráči házet:

$$útočník = \min\{3, počet\_jednotek\}$$
$$obránce = \min\{2, počet\_jednotek\}$$

Útočník tedy hází maximálně třemi kostkami, obránce dvěma.

Průběh boje Obě strany hodí kostkami a porovnají se výsledky. Pokud útočníkovo nejvyšší číslo převyšuje obráncovo nejvyšší, obránce si ihned odstraní z hracího plánu jednu jednotku z napadeného území. V případě shody nebo vyššího čísla obránce ztratí útočník jednu útočící jednotku. Pokud házeli hráči více než jednou kostkou, postupuje se obdobně u druhých nejvyšších čísel.

Útočník se rozhodne, zda bude pokračovat nebo revoluci přeruší (což by většinou nemělo smysl, ztratil by útočné jednotky bez možnosti náhrady). V případě, že se rozhodne útočit dál, se opakuje proces popsaný v odstavci *Průběh boje*. Pokud útočník vyčerpá své jednotky, může zahrát nějakou další kartu revoluce nebo přejít k další fázi tahu. Pokud dojdou jednotky obránci, území získá útočící hráč, osadí jej všemi jednotkami, které přežily revoluci, a vezme si novou kartu revoluce, kterou bude moci zahrát v dalších kolech.

1.4.3 Fáze útoku (attack)

Dostáváme se k nejdůležitější fázi – útoku. Hráč vybere území, odkud chce vést útok, a cíl útoku. Aby hráč mohl zaútočit na nějaké území, musí být splněny následující podmínky:

- Území, odkud útočí, musí sousedit s napadenou zemí.
- Území, odkud útočí, musí být osídleno více než jedním vojákem.

Následně se určí, kolika kostkami mohou hráči házet, stejným způsobem jako v revoluční fázi. Útočník i obránce mohou házet méně kostkami, než jim bylo určeno, nejméně však jednou. Útok samotný se vyhodnocuje jako ve fázi revoluce. Odlišnost od revoluce je taková, že v zemi, odkud je útok veden, musí vždy zůstat alespoň jeden voják. Ten se tedy útoku nesmí zúčastnit a není započítáván do počtu dostupných kostek. Útok končí buď úspěšným dobytím (v obráncově území nezůstane žádná jednotka), nebo vyčerpáním útočících jednotek (v útočníkově území zůstane jen jedna jednotka). Útočící hráč může kdykoliv s útokem přestat, pokud tak uzná za vhodné. V případě, že je útočník úspěšný,

přesune vojáky, kteří útočili (stejný počet jako kostek před posledním hodem), do dobytého území a získá novou kartu revoluce.

Počet útoků během jednoho tahu není omezen. Ve chvíli, kdy hráč nechce provádět žádné další útoky, se tah posune do fáze pohybu.

1.4.4 Fáze pohybu (move)

Zde může hráč přesunout jednotky mezi dvěma svými územími, která spolu sousedí. Zvolí si, kolik vojáků přesune, avšak stále platí, že v každém území musí zůstat alespoň jeden voják. Jednotky lze přesunout pouze jednou za kolo.

1.4.5 Konec tahu (end turn)

Hráč ukončí svůj tah. Pokud bude mít více než 5 karet revoluce z minulých kol, musí vyhodit přebytečné karty na odhazovací balíček. Nyní je na tahu další hráč.

1.5 Pravidla původní hry

Nyní uvedeme hlavní rysy pravidel původní hry Risk a jejich odlišnosti od těch upravených. Originálními pravidly se víceméně řídí podobné implementace popsané v kapitole 5.

1.5.1 Začátek hry

Na začátku hry si každý hráč zvolí libovolné neobsazené území a vloží do něj jednoho vojáka. Hráči se takto střídají, dokud existují volná území. Vidíme tedy, že v původním Risku nejsou neutrální jednotky.

1.5.2 Posilování a pohyb

Obě fáze jsou totožné jako v modifikovaných pravidlech.

1.5.3 Revoluční karty

Karty slouží jako prostředek k získání bonusových jednotek. Pokud hráč vlastní 3 karty, jež mají všechny stejnou nebo každá jinou sílu, může je směnit za jednotky, které použije v posilování. Počet jednotek, jež lze tímto způsobem získat, je fixní, nebo se zvyšuje po každém obchodu.

1.5.4 Útok

Průběh útoku je stejný jako v tomto projektu. Liší se ale přesunem jednotek po útoku. Zatímco v naší verzi se přesune pevně daný počet, v původní verzi si lze vybrat. Útočník přesune minimálně tolik jednotek, s kolika útočil (počet kostek před posledním hodem), a maximálně všechny dostupné jednotky.

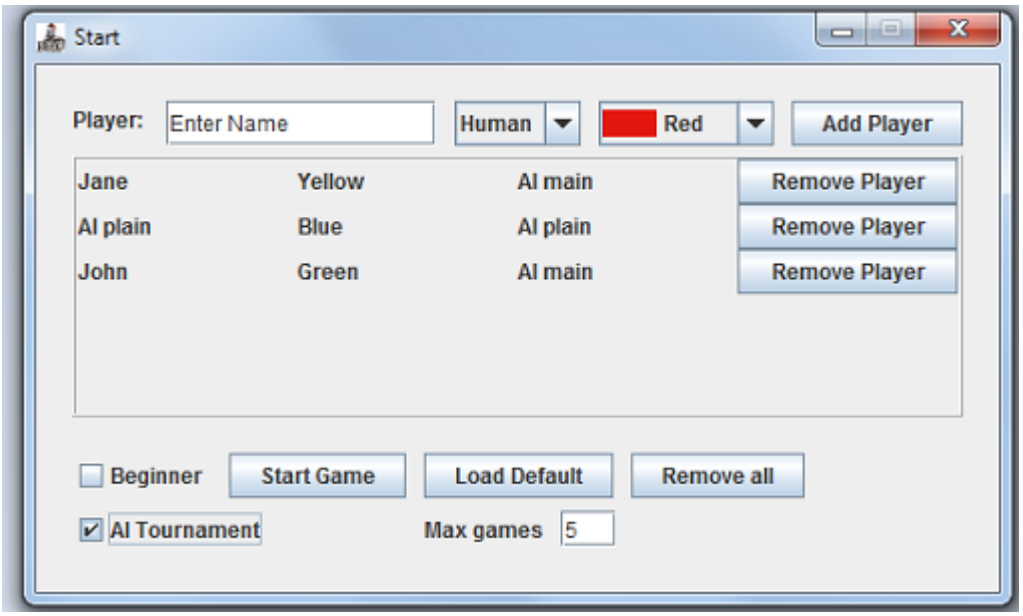
2. Uživatelské rozhraní a ovládání

2.1 Instalace a spuštění

Instalace je snadná. Stačí hru uložit na disk a spustit, případně pouze spustit přímo z CD. K bezproblémovému běhu aplikace je třeba mít nainstalovanou Javu verze 7. Po spuštění souboru Risk.jar ve složce *Distribuce Risk* se objeví úvodní obrazovka.

2.2 Úvodní obrazovka

Jakmile spustíme program, objeví se úvodní obrazovka (viz Obrázek 2.1). Nyní si projdeme prvotní nastavení před samotnou hrou.



Obrázek 2.1: Úvodní obrazovka

2.2.1 Přidání hráče

Než hra odstartuje, musíme nastavit, jací hráči se zúčastní. Vyplníme jméno v příslušné kolonce (vlevo nahoře podle obrázku 2.1). Dále vybereme typ hráče, máme tři možnosti:

1. Human — hráč ovládaný člověkem
2. AI main — hráč ovládaný počítačem, hlavní AI využívající prohledávání
3. AI plain — přímočarý umělý hráč; hraje o něco hůře než předchozí.

Zbývá vybrat barvu (každý hráč musí mít samozřejmě jinou). Následně může být hráč přidán klepnutím na tlačítko *Add Player*. Posléze se objeví v tabulce nový hráč (viz Obrázek 2.1). Tlačítkem *Load Default* načteme předuložené hráče. Je možno přidat až 5 hráčů.

2.2.2 Odebraní hráče

Pro odebrání hráče jednoduše stiskneme tlačítko *Remove Player* u příslušného hráče. Chceme-li odebrat všechny hráče, stačí kliknout na tlačítko *Remove all* v pravé dolní části obrazovky.

2.2.3 Nápověda

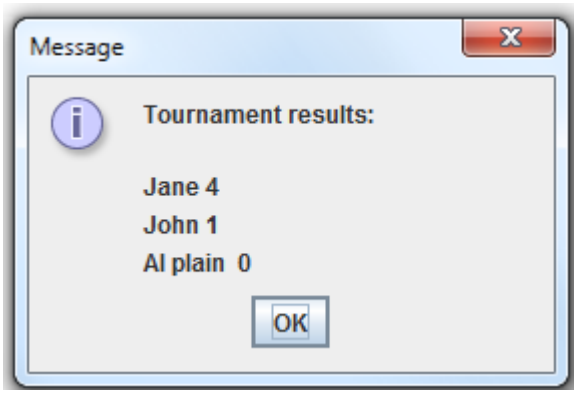
Zaškrtnutím volby *Beginner* zapneme začátečnický režim a zajistíme, že budeme v průběhu hry dostávat rady o pravidlech hry a ovládání uživatelského rozhraní. Toto nastavení můžeme kdykoliv vypnout/zapnout i ve hře pomocí menu: *Game/Beginner*.

2.2.4 Start hry

Jakmile je vše připraveno, tlačítko *Start Game* spustí novou hru. Je možné, že bude třeba počkat několik sekund, než program načte potřebné soubory.

2.2.5 Turnaj AI

Hru je možné spustit i bez lidských hráčů. Pokud nejsou ve hře žádní lidští hráči, zpřístupní se zaškrtávací tlačítko *AI Tournament*, které umožňuje hrát turnaj mezi AI hráči (viz obrázek 2.1). Po jeho zaškrtnutí se zobrazí možnost vyplnit pole *Max games*. Můžeme vyplnit hodnotu mezi 1-50. Program potom odehraje tolik her, kolik jsme vybrali, a na závěr zobrazí pořadí hráčů v turnaji (viz Obrázek 2.2).



Obrázek 2.2: Výsledek turnaje

2.3 Ovládání hry

2.3.1 Hlavní obrazovka

Obrázek 2.3: Hlavní obrazovka

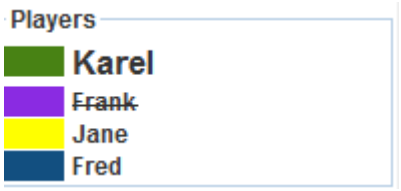
Obrázek 2.3 ukazuje, jak vypadá hlavní obrazovka. Nyní si rozebereme jednotlivé části zvlášť.

Prvky hlavní obrazovky

Mapa Ukazuje stav hry. Jednotlivé státy jsou odděleny černou hranicí. Sousední státy, které odděluje moře, poznáme podle bílých průhledných spojnic. Čísla uvnitř území značí počet vojáků. Každé teritorium je vybarveno podle vlastníka. Území obsazené neutrálními jednotkami má šedou barvu. Hlavní města jsou označena kruhem barvy příslušného hráče. Pokud hráč ztratí své hlavní město, kruh bude mít stále stejnou barvu, území se však obarví podle dobyvatele.

Ukazatel teritoria V pravé horní části obrazovky se nachází ukazatel, který reaguje na pohyb myši. V případě, že se myš pohybuje nad nějakým územím, nápis ukazuje název tohoto území a zároveň počet jednotek. To je užitečné, zejména když hrajeme na menším monitoru a máme potíže přečíst počet vojáků nebo si nejsme jisti, zda míříme na to či ono území. Mimo státy nápis ukazuje *No Country*.

Přehled hráčů Tento panel určuje pořadí hráčů, detail na Obrázku 2.4.



Obrázek 2.4: Panel hráčů

Zvýrazněný hráč, uvedený nahoře, je právě na tahu. Po odehrání se ikona s vrchním hráčem přesune na konec seznamu a na řadě je další hráč. Zničený hráč je v panelu přeškrtnut rovnou čarou (viz Obrázek 2.4.). Klikneme-li na kteréhokoliv hráče, ukáží se nám o něm základní informace (viz Obrázek 2.5.). Tyto informace usnadňují práci, když například chceme vědět, kolik má soupeř jednotek za kolo, ale nejsme ochotni vyvinout úsilí to spočítat.

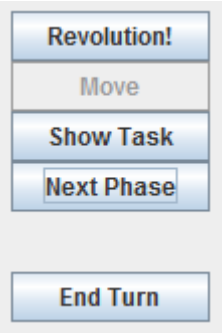
Obrázek 2.5: Informace o hráči

Informace o hráči jsou popořadě tyto:

- Jméno hráče
- Počet starých karet
- Počet karet získaných v tomto kole; po ukončení tahu se tyto karty přesunou do starých.
- Počet ovládaných zemí
- Počet ovládaných kontinentů
- Počet jednotek, které má k dispozici na začátku tahu.
- Hráč, který na vybraného hráče naposledy zaútočil a zabil mu alespoň jednu jednotku.

Popisky Pod panelem s hráči se nachází popisky, které se mění podle fáze hry. Dále ukazují výběr útočníka/obránce, více v sekcích 2.3.2 až 2.3.7 o jednotlivých fázích.

Tlačítka O něco níže nalezneme ovládací tlačítka, detail viz Obrázek 2.6.



Obrázek 2.6: Panel tlačítek

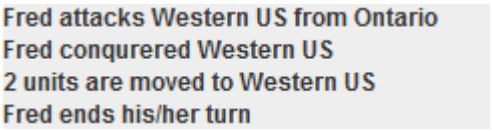
Výčet tlačítek:

- Revolution — v kterékoliv fázi tahu se můžeme podívat na své karty. Hrát je však můžeme pouze v revoluční fázi.
- Tlačítko akce má dvě role: pohyb a útok, je aktivní pouze ve stejnojmenných fázích, jinak je stisknout nelze.
- Show Task — dialogové okno (Obrázek 2.7) nám připomene, jaký máme tajný úkol.

Obrázek 2.7: Tajný úkol

- Next Phase — přesune tah do následující fáze.
- End Turn — ukončí tah a nechá hrát dalšího hráče.

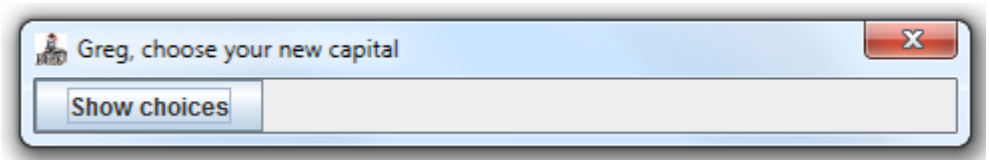
Panel zpráv Ve spodní části obrazovky je panel, kam se ukládají všechny důležité události hry. Pokud nám tedy něco unikne, můžeme cokoliv zpětně dohledat. Ukázku naleznete na obrázku 2.8.



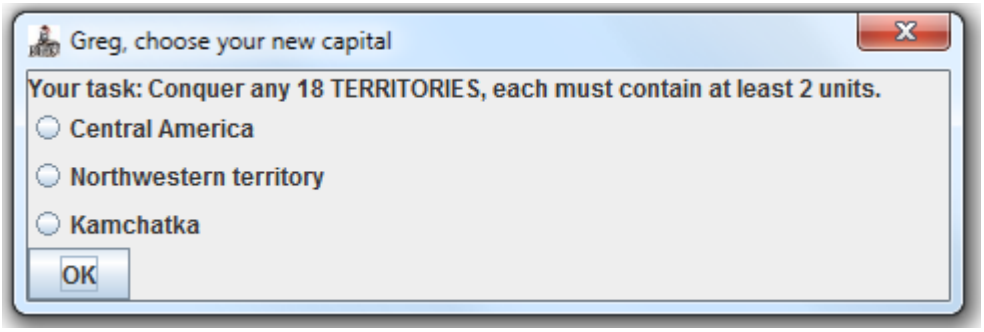
Obrázek 2.8: Panel zpráv

2.3.2 Výběr hlavního města

V úvodní části hry nás uvítá hlavní obrazovka s dialogovým oknem (Obrázek 2.9). Nachází se na něm jediné tlačítko *Show choices*. Ostatní obsah zůstane skryt, dokud tlačítko nestiskneme. Tímto je v případě hry více lidí na jednom počítači zajištěno, že hráči vědí, kdy mají odvrátit zrak. Po stisku zjistíme, jaký je náš tajný úkol, vybereme si z nabídky své hlavní město a potvrdíme tlačítkem *OK*. Následně vybrané území získá naši barvu a 7 jednotek. Takto se vystřídají všichni hráči a hra dále pokračuje tahem prvního.



(a) Skrytý



(b) Odhalený

Obrázek 2.9: Výběr hlavního města

2.3.3 Posilovací fáze

Popisek v pravé části obrazovky nás informuje, že je nyní posilovací fáze *(Phase: Reinforce)*. Další ukazuje, kolik nám zbývá přiřadit jednotek *(Units left)*. Pro přidání jednotky jednoduše klepneme na kýžené území. Jednotky lze přiřazovat pouze do vlastněného státu. Levé tlačítko přidá jednoho vojáka, pravé deset, respektive zbytek menší než deset.

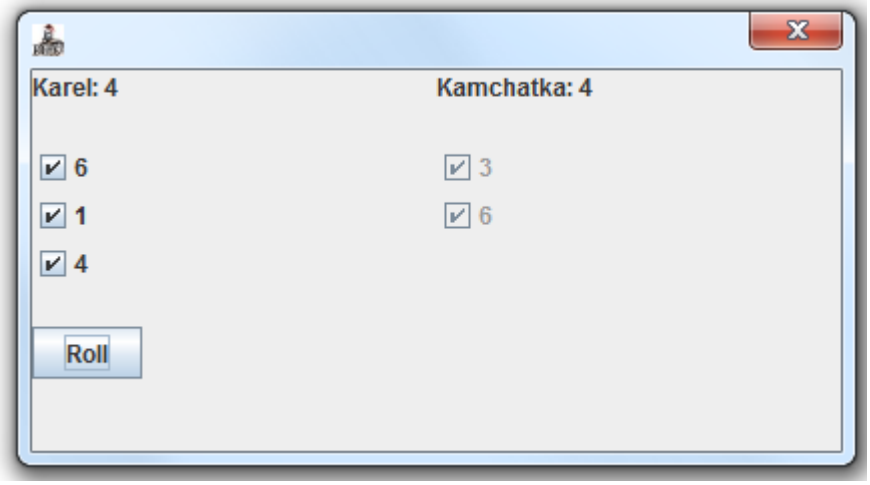
2.3.4 Revoluční fáze

Pokud chceme hrát revoluční karty, či si je jen prohlédnout, stiskneme tlačítko *Revolution*. Jak ukazuje obrázek 2.10, v horní části jsou karty, které můžeme odehrát v tomto tahu. Pod nimi jsou ty, které jsme získali v současném kole, a budeme je tedy moci zahrát až příště.

Obrázek 2.10: Revoluční karty

Každá karta/tlačítko má barevné orámování odpovídající barvě hráče, který území vlastní. Jak jsme již uvedli výše, šedá barva značí neutrální jednotky. Na každé kartě je kromě názvu ještě číslovka. Ta značí sílu karty. Kupříkladu, pokud je na tahu zelený hráč z obrázku a zahraje kartu *Irkutsk*, posílí si Irkutsko třemi vojáky. Pokud zahraje *Indonesii*, napadne fialovému hráči stejnojmenné území pěti vojáky. Karty se zahrají kliknutím. Okno lze zavřít pomocí tlačítka *Done* či křížkem.

Revoluční útok Po vyvolání revolučního útoku se objeví okno (Obrázek 2.11).

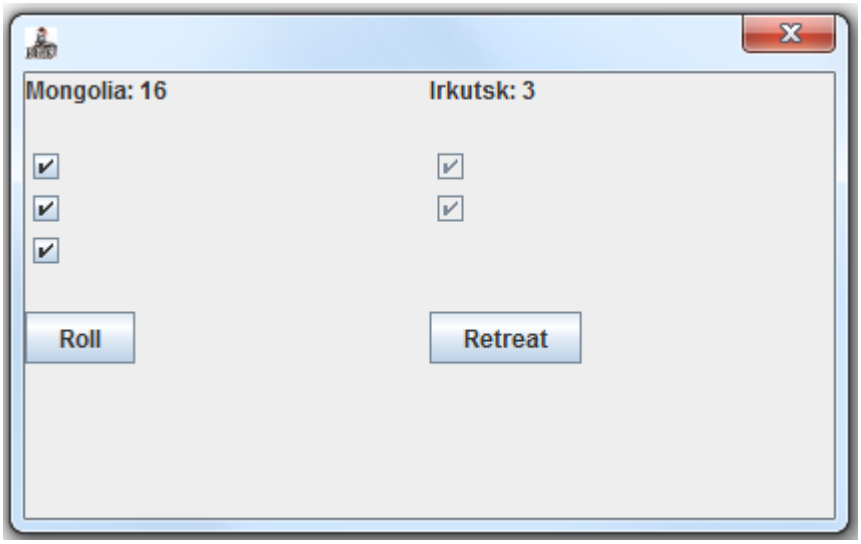


Obrázek 2.11: Revoluční útok

Levá část je věnována útočníkovi, pravá obránci. V horní části vidíme, kolik má kdo k dispozici jednotek. Ve střední části se nacházejí hrací kostky reprezentované zaškrtávacími tlačítky. Obránce i útočník si vybere, kolika kostkami bude házet. V situaci na obrázku jsou obráncovy kostky zašedlé. Je to proto, že Kamčatku ovládá počítač. Všimneme si, že útočník hodil čísla 6, 1, 4 a obránce 3, 6. Podle pravidla popsaného v sekci 1.4.2, zemřela útočícímu hráči jedna jednotka ($6 \leq 6$) a obránci rovněž ($4 > 3$). Po zvolení počtu kostek útočník zahájí hod tlačítkem *Roll*. Pokud již nějakou kostku nelze použít, zaškrtávací tlačítko se stane neaktivním. Hází se tak dlouho, dokud revolucionář nevyplýtvá všechny jednotky, nebo nedobude napadené území.

2.3.5 Útočná fáze

Chceme-li zaútočit, je třeba vybrat území, z něhož útok povedeme, a jeho cíl. V popiscích označeno jako *Attacker* a *Defender*. Vybíráme kliknutím levého tlačítka myši na cílové území. Po úspěšném zadání se nápisy ve výše zmíněném popisku příslušně aktualizují. Chceme-li odznačit naposledy vybrané území, klikneme na ně pravým tlačítkem. Za předpokladu, že chceme odznačit cíl i výchozí bod útoku, klikneme pravým tlačítkem kamkoli jinam na mapu. Máme-li vybráno, pak stiskem tlačítka akce (v této fázi tahu s nápisem *Attack!*) vyvoláme okno útoku (Obrázek 2.12).



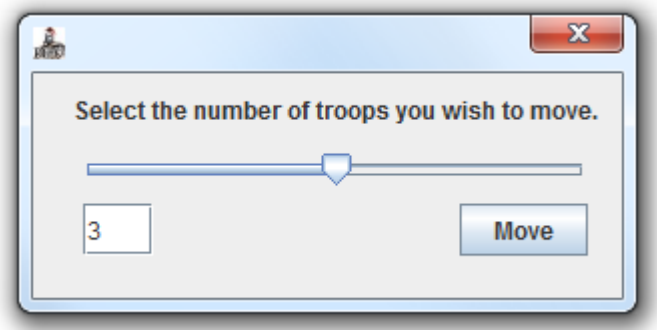
Obrázek 2.12: Útok

Toto okno je velice podobné revolučnímu oknu z Obrázku 2.11. Liší se pouze tím, že v levé horní části je název útočícího území a počet jednotek k dispozici je o jedna menší než skutečný počet vojáků (všude musí vždy zůstat minimálně jeden). Další rozdíl skýtá tlačítko *Retreat*. Slouží k okamžitému přerušení boje. Pokud ho člověk použije, může útočit dále do jiných území nebo přejít k fázi pohybu. Dobyde-li hráč území, bude mu tam přesunuto tolik jednotek, kolika kostkami házel.

Poznámka Tlačítko *Roll* u obyčejného i revolučního útoku vždy mačká jen útočník. Pokud útočí počítač proti počítači, uživatel čeká, umělí hráči se o vše postarají sami.

2.3.6 Fáze pohybu

Ovládaní je podobné jako u útočné fáze. Kliknutím vybereme, odkud a kam chceme přesunout jednotky, tato území se opět zobrazí v popisku v pravé části obrazovky. Nesmíme zapomenout, že lze přesouvat jen v rámci sousedních vlastněných území. Jsme-li spokojeni s výběrem, stiskneme tlačítko *Move* a v okně vyobrazeném níže (Obrázek 2.13) vybereme počet, který chceme přesunout, a potvrdíme tlačítkem *Move*.



Obrázek 2.13: Pohyb

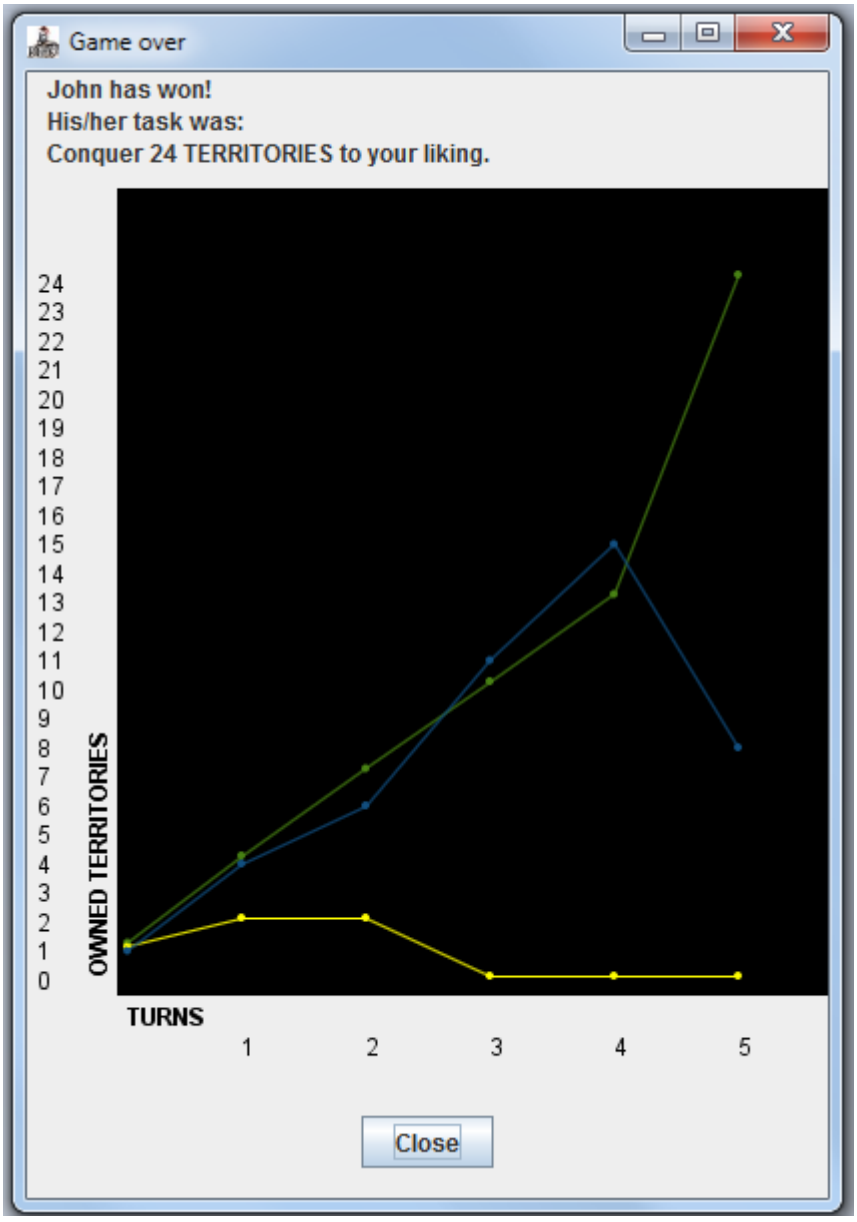
2.3.7 Ukončení tahu

Tlačítko *End Turn* ukončí náš tah a předá otěže dalšímu hráči. Pokud však počet karet, které jsme získali v předchozích tazích a dosud nepoužili, převyšuje 5, je nutno některé z těchto karet vyhodit. Automaticky se vyvolá okno z obrázku 2.10 a budeme vyzváni k odhození karet. Učiníme tak klepnutím na příslušnou kartu. Až zredukujeme počet starých karet na 5 či méně, můžeme tah ukončit.

2.3.8 Konec hry

Po ukončení hry se zobrazí časová linie znázorňující vývoj hry (viz Obrázek 2.14). Osa x ukazuje počet kol a osa y počet vlastněných teritorií na konci tahu každého hráče. Všichni hráči mají v grafu křivku své barvy. V horní části obrazovky se dočteme, kdo vyhrál a jaký měl splnit úkol.

V případě, že se hrál AI turnaj, zobrazí se místo časové osy výsledek turnaje (Obrázek 2.2).



Obrázek 2.14: Časová osa

2.4 Menu

V této části rozebereme rozbalovací nabídku programu.

2.4.1 Možnosti mapy

V nabídce *Map* nalezneme následující možnosti:

- Show Names — zaškrtnutím této možnosti zajistíme zobrazení jmen území na mapě.
- Show Transitions — je-li tato možnost označena, na mapě jsou zobrazeny spojnice mezi územími.
- Show owned territories — v tomto zobrazení uvidíme na mapě území obarvená podle vlastníků.
- Show continents — tato možnost nám ukáže barevně odlišené kontinenty:
  - North America: bílá
  - South America: žlutá
  - Europe: zelená
  - Asia: šedá
  - Australia: tyrkysová
  - Africa: oranžová

2.4.2 Možnosti hry

Položka *Game* skýtá tyto možnosti:

- New Game — vyvolá úvodní obrazovku s nastavením hráčů, můžeme začít novou hru.
- Load Game — načte hru ze souboru s příponou *.rsk*.
- Save Game — uloží hru pod názvem, který si zvolíme.
- Beginner — je-li zvoleno, v každé fázi tahu dostaneme rady, jak si počínat.
- Exit — ukončí program.

2.4.3 Možnosti okna

V menu *Window* volíme velikost mapy. Od nejmenšího (*Nearly invisible*) po největší (*Larger*). Pokud máme příliš malou obrazovku, program nás upozorní. Program si zapamatuje zvolenou velikost a v příštích bězích ji tak automaticky nastaví.

3. Implementace

Celý projekt je implementován v jazyce JAVA a používá standardních knihoven. Grafické rozhraní využívá Swing Toolkitu. Níže jsou uvedeny všechny balíčky projektu a některé zajímavé třídy.

3.1 Soubory projektu

Součástí projektu jsou kromě zdrojových kódů také soubory s nastavením, uložené hry a tak podobně. Níže si jednotlivé soubory rozebereme.

3.1.1 Mapa

Ve složce *img* se nachází čtyři obrázky související s mapou.

- `map.png` — hlavní obrázek mapy, který bude vždy zobrazen. Území budou podle volby uživatele vybarvena buď podle barvy hráče, jenž je vlastní, či podle barvy kontinentu, kam náleží.
- `names.png` — obrázek, jenž se skládá pouze z nápisů s názvy území. Tyto nápisy jsou v obrázku umístěny na stejných pozicích jako území na herním plánu.
- `borders.png` — na obrázku jsou čarami znázorněny spojnice mezi sousedními zeměmi, jež jsou odděleny mořem.
- `hidden.png` — používá se při inicializaci. Každé území je zde vyplněno jinou barvou. Algoritmus dostane mapování indexu území na barvu a vytvoří dvourozměrné pole bytů, které vyplní tak, že na pozici x,y bude index území nebo -1 pro souřadnice mimo pevninu. Toto pole se v programu používá pro zachytávání událostí myši na herním plánu. Výhoda tohoto přístupu tkví v jednoduché identifikaci teritoria, na něž uživatel klikl nebo najel myší.

Prvé tři obrázky jsou odděleny, aby si uživatel mohl vybrat, co všechno si nechá zobrazit. V případě, že chce mít uživatel zobrazeny názvy teritorií i přechody mezi nimi, program obrázky nakreslí přes sebe.

3.1.2 Logy

Ve složce *Logs* nalezneme soubor log, který v jednoduché formě ukazuje poslední vygenerovaný prohledávací strom při hledání tahu AI hráče. Každý uzel obsahuje název akce vedoucí do daného stavu a hodnotu stavu z podstromu. Podle fáze tahu mohou uzly obsahovat ještě nějaké jiné informace, kupříkladu pravděpodobnost úspěšného hodu kostkou nebo jméno hráče. Kořen stromu se nachází na řádku co nejvíce vlevo, uzly s přibývající hloubkou jsou odsazeny vpravo. Log sloužil pro ladění programu a na funkčnost jako takovou nemá vliv.

3.1.3 Uložené hry

Uložené hry se typicky uchovávají ve složce *Saved Games*. Pro ukládání byla zvolena serializace herních objektů.

3.1.4 Data

V této složce jsou soubory s informacemi, které je nutné načíst před začátkem hry.

- `armyCoord` — každý řádek tohoto souboru obsahuje ID území a dvojici souřadnic. Na tyto souřadnice se v mapě vykresluje počet jednotek daného území.
- `colors` — na každém řádku je ID území a celočíselná hodnota jeho barvy v obrázku *hidden.png*.
- `init` — zde jsou veškerá data o kontinentech, územích, jejich sousedech a o tajných misích. Program soubor přečte a na základě dat vytvoří objekty pro hru.
- `Probabilities` — uložená instance třídy, která zabaluje pomocnou tabulku pro odhad výsledků boje.

3.2 Architektura

V této sekci jsou vypsány všechny balíčky projektu a jejich účel.

3.2.1 Balíček AI

Zde se nachází obsáhlá třída AI, která obstarává hledání tahů pro AI hráče. Dále jsou zde třídy reprezentující stavy hry v prohledávacím stromu a další podpůrné třídy. Dále skýtá balíček s rozhraním pro přímočarého hráče a jednu jeho implementaci. Do hloubky budou algoritmy umělé inteligence popsány ve čtvrté kapitole (včetně implementace). Zejména třída *State* a její potomci.

Třída AI Jako důležité metody této třídy lze uvést například tyto:

- `CreateState` — pomocí stavu hry vytvoří odpovídající stav pro prohledávací strom.
- `getMove` — vybere nejlépe ohodnocený stav hry a vrátí tah, který do něj může vést. V počátcích programu se algoritmus prohledávání řídil pouze hloubkou stromu, což se neukázalo býti vhodným přístupem. Důvody a alternativní řešení budou uvedeny v následující kapitole.
- `maxValue` — tato rekurzivní metoda vygeneruje a ohodnotí prohledávací strom a vrátí nejlepší hodnotu z listu pro hráče, který je právě na tahu.
- `nextState` — třída AI si drží kořen prohledávacího stromu a tato metoda se ve stromě posune o úroveň níž podle vybrané akce a vybere nový kořen. Pokud akce byla jiná než hod kostkou, dá se strom použít i pro další tahy bez nového generování.

Balíček StraightforwardAI Skládá se z rozhraní, které bylo převzato od společnosti Sillysoft[7]. Bylo upraveno pro tuto implementaci jednak pro různá pravidla, jednak pro odlišný návrh hry. Rozhraní má jednu implementaci a to třídu Angry, jejíž principy rovněž pochází ze stránek společnosti Sillysoft[7].

Třída LongRunningTask Protože hledání vhodného tahu může být velice dlouhé (složitost metody maxValue je exponenciální), byla naprogramována třída LongRunningTask, která implementuje rozhraní Callable. Zde probíhá výpočet tahu umělého hráče. Výše uvedená metoda getMove ji zavolá s předem definovaným časovým limitem. Pokud algoritmus překročí časový limit, výpočet je přerušen a je zavolána rychlá metoda pro výběr tahu, která funguje následovně. Pokud hráč může v tomto tahu házet kostkou, hází, jinak je vybrána akce pomocí prohledávání pouze do hloubky 2.

3.2.2 Balíček RiskAction

Balíček obsahuje hlavně rozhraní *IRiskAction*, které reprezentuje možné akce. Pro každou akci hratelnou ve fázích tahu existuje třída, která implementuje výše zmíněné rozhraní. S těmito akcemi pracuje GUI i umělá inteligence. Dále obsahuje pomocné třídy pro různé akce. Tyto pomocné třídy jsou od *IRiskAction* odvozeny. Zde je přehled tříd:

- `IRiskAction` — skládá se ze statických proměnných, které představují možné typy akcí. Dále ze dvou abstraktních metod:
  - `getType` — každá akce vrací identifikátor svého typu, přičemž každý tento typ odpovídá jedné odvozené třídě.
  - `print` — vytiskne informace o akci.
- `RiskActionAddSoldiers` — akce, která přidá vojáky v posilovací fázi. Obsahuje pole pomocných struktur *AddSoldiersPair*, která informuje kam a kolik vojáků přidat.
- `RiskActionAddRev` a `RiskActionBeginRev` — obě akce jsou hratelné v revoluční fázi a mají stejný parametr — index teritoria, ke kterému patří revoluční karta. Prvně jmenovaná třída reprezentuje revoluční posilování, zatímco druhá revoluční útok.
- `RiskActionSuperRevolution` — vytvoří stav po zahrání všech karet revoluce najednou.
- `RiskActionBeginAttack` — vyvolá útok. Parametr *AttackInfo* informuje o cíli a původci útoku ve formě indexů příslušných států.
- `RiskActionRollAI` — hod kostkou v prohledávacím stromu. Jedná se o posloupnost hodů končící dobytím, nebo vyčerpáním jednotek útočníka. AI tedy předpokládá pouze dva možné výstupy: dobytí, vyčerpání útočících jednotek. Booleovský parametr značí předpokládaný výsledek hodu. Třída má potomka *RiskActionRollDice*, jenž se používá přímo ve hře, navíc má

parametry, kolika kostkami hráči házejí. Potomek nic neodhaduje, provede hod tak, jak dostane předepsáno.

- `RiskActionRetreat` — ukončí útok.
- `RiskActionSkipPhase` — přesune tah do další fáze.
- `RiskActionMove` — přesune jednotky na konci kola. Parametrem je instance pomocné třídy *MoveActionInfo* obsahující indexy území, mezi kterými se přesun provede, a počet přesouvaných jednotek.

3.2.3 Balíček Editor

Tento balíček obsahuje jedinou třídu a to Editor, díky kterému je možné uloženou hru pozměnit. Usnadňuje simulaci různých herních situací. Není však úplně odladěn, proto chybí jeho popis v uživatelské sekci. Není doporučováno jej používat.

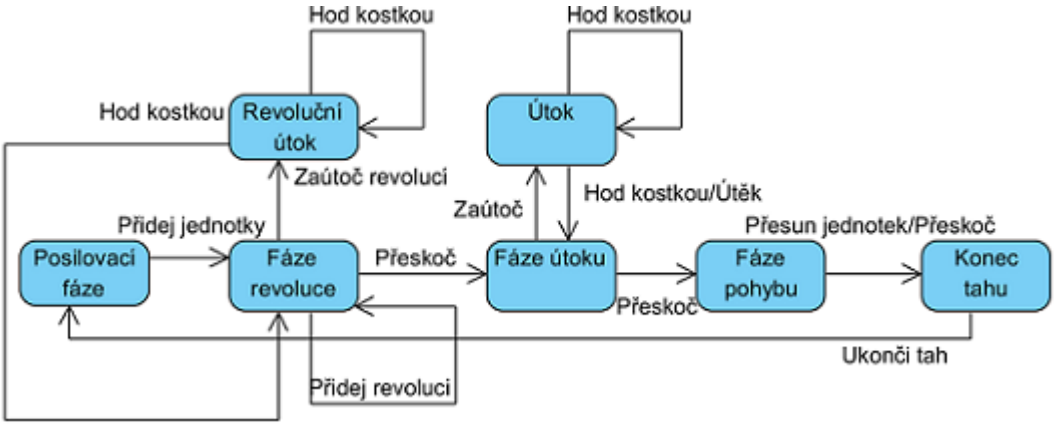
3.2.4 Balíček risk

Nejdůležitější balíček, obsahuje základní herní objekty a mechanismy. Při návrhu projektu byla snaha o oddělení jádra hry, prezentace a umělé inteligence.

Třídy

- `GameHistory` — uchovává historii probíhající hry. Pro každého hráče si pamatuje počet území v každém kole. Pokaždé, když hráč ukončí svůj tah, aktualizuje se počet jeho teritorií v probíhajícím kole. Data z této třídy se používají k vykreslování časové osy na konci hry.
- `Territory` — každé území má svůj index, vlastníka a index kontinentu, kam patří.
- `Continent` — kontinent disponuje seznamem indexů všech území, která do něj náleží. Také má případně vlastníka, pokud někdo ovládá všechna území tohoto kontinentu.
- `Player` — důležitá data o hráči, jeho index, typ, seznam indexů vlastněných teritorií, kontinentů a karet revoluce.
- `Task` — tato třída představuje tajný úkol, jehož splněním lze vyhrát hru. Pokud již úkol nelze splnit, automaticky se změní na předdefinovaný sekundární.
- `Game` — zde se nalézají veškeré údaje o herním stavu. Tato třída má v kolekcích uložené všechny hráče, území, kontinenty, balíček karet a další. Původně byla v návrhu i třída pro karty revoluce. Později však byla vyňata, protože stačí mít index území, ke kterému patří, a u území atribut síly karty. Hra se chová jako konečný automat. Na obrázku 3.1 je znázorněn takový automat pro tah hráče. Například na stavech *Revoluční útok* a *Útok* vidíme, že se chová nedeterministicky. Pro zjednodušení v diagramu není znázorněn koncový stav. V závislosti na tajném úkolu lze teoreticky koncového stavu dosáhnout ze všech stavů. Stavy tohoto automatu projde každý

hráč ve svém tahu. Automat však nepokrývá všechny stavy. Chybí například volba hlavního města, ke které dochází pouze na začátku hry, nebo fáze pohybu jednotek po úspěšném dobytí, která byla pro jednoduchost připojena k úspěšnému hodu a vede do stavu *Útok*.



Obrázek 3.1: Přechody mezi stavy hry

- `Attack` — vnitřní třída, která je součástí *Game*. Slouží k vedení a vyhodnocování útoků. Potomek *RevolutionAttack* má na starost revoluční útoky. Díky polymorfismu není třeba rozlišovat mezi druhy útoků, když se vyhodnocuje hod kostkou.
- `playerStruct` — jméno hráče, typ hráče, index barvy. Slouží jako definice hráčů na úvodní obrazovce a pro jejich ukládání a načítání.
- `PlayerSettings` — tato třída definuje metody pro ukládání a načítání lokálního nastavení pomocí třídy *Preferences*. Jedná se například o metody pro načítání hráčů, nastavení začátečnického módu, velikost mapy.

3.2.5 Balíček GUI a map

Zde nalezneme třídy pro uživatelské grafické rozhraní a potomky různých komponent, například panel pro zobrazení mapy.

MainRiskForm Nejdůležitější z těchto tříd je MainRiskForm, která obstarává grafické uživatelské rozhraní v průběhu hry. Stará se nejen o vykreslování mapy a obsluhování herních událostí, ale také o předávání řízení mezi lidskými a umělými hráči.

Dialogová okna Dále jsou zde třídy *RevAttackDialog*, resp. *AttackDialog*, které slouží ke zobrazování průběhu revolučních útoků, resp. útoků. Tyto dialogy se graficky liší pouze ve skutečnosti, že revoluční dialog nemá tlačítko pro ústup. Proto byl vytvořen společný předek *CommonAtkDialog*. Umělý hráč hází v revolučním útoku automaticky bez prohledávání stromu. V obou dialozích se v konstruktoru určí typ obránce i útočníka. Tlačítko *Roll* podle toho bude provádět různý kód. Možnosti jsou tyto:

- AI proti AI — uživatel je pouze v roli diváka a ovládání GUI mu není přístupné. Ve smyčce se útočná AI rozhoduje, jestli bude házet, či se stáhne. V prvním případě se po hodu vždy aktualizují grafické komponenty a ve druhém se dialog uzavře.
- Člověk proti člověku/AI — hru ovládá útočník. Obránce hází kostkou vždy poté, co hodí útočník, takže na straně obránce není třeba žádného rozhodování.
- AI proti člověku — tlačítko *Roll* sice ovládá lidský hráč, ale lze ho stisknout až potom, co se AI rozhodne, že bude házet. Program to řeší tak, že si vygeneruje strom a podívá se na svůj tah, ale nezahraje ho. Pokud se rozhodne k hodu, učiní tak až po uživatelově stisku tlačítka. V případě útěku není od uživatele vyžadována ani očekávána žádná akce, dialog se automaticky zavře.

Předávání řízení hry Při začátku hry nebo při jejím načtení třída MainRiskForm zkontroluje, jaký typ hráče má táhnout. Pokud jde o člověka, odemkne mu ovládání hry a nechá ho hrát, dokud neukončí tah. Pokaždé, když se změní hráč, se opět provede kontrola hráče. V případě umělého hráče se spustí metoda *aiMove()*.

Průběh tahu umělého hráče Jakmile se k tahu dostane AI hráč, metoda aiMove() vytvoří instanci třídy SwingWorker, která umožní spuštění úlohy na pozadí, a tím předejde přetížení vlákna na obsluhu událostí. Algoritmus začne procházet smyčku, kterou neukončí, dokud hra nedospěje do fáze konce tahu nebo konce hry. V prvním případě se automaticky ukončí tah, ve druhém se GUI odemkne pro lidského hráče po zobrazení vítěze hry a grafu s časovou osou. Ve smyčce se nejprve provede jeden tah. Podle toho, zda jde o prohledávací či přímočarou AI, se buď vygeneruje strom, či se najde tah přes příslušného agenta. Poté se vlákno uspí na dobu definovanou ve třídě *Game*, aby se hra dala plynule sledovat. V případě, že AI vybere útok, vyvolá se dialogové okno s útokem. V tomto GUI okně bude probíhat nová smyčka s hledáním tahů. Původní smyčka se v průběhu aktivního dialogu uzamkne. Po skončení útoku se opět odemkne a hráč táhne dál.

3.2.6 Balíček support

V tomto balíčku se nacházejí různé podpůrné třídy: Generická halda, třída na kreslení jednoduchého grafu, třída na odhad pravděpodobnosti vítězství v boji a další. Některé byly používány pouze na začátku vývoje projektu.

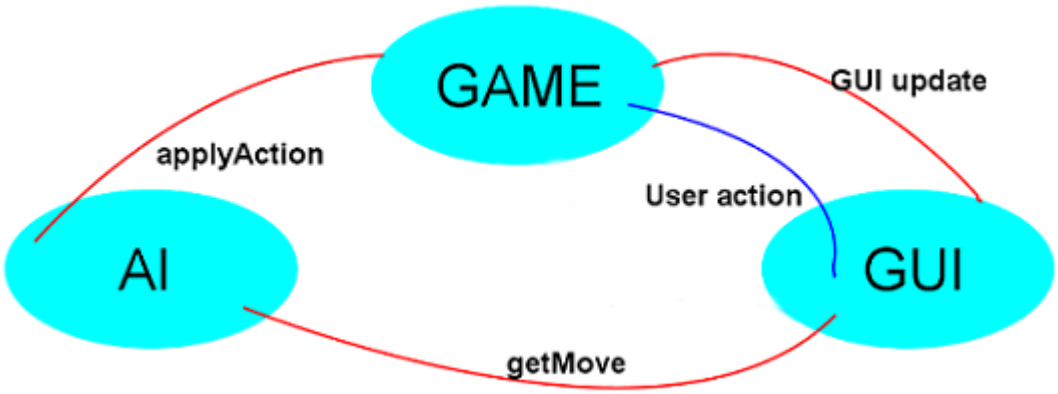
Třída Probabilities Tato pomocná třída obsahuje tabulku, která je používána algoritmy AI pro odhad pravděpodobnosti vítězství boje. Nechť souřadnice x značí počet jednotek útočníka a y počet jednotek obránce. Pak se v tabulce na pozici x,y nachází instance třídy *probTriplet* o třech údajích:

1. Pravděpodobnost, že útočník vyhraje a dobude napadené území.
2. Počet jednotek útočníka, které průměrně přežijí.
3. Počet jednotek obránce, které průměrně přežijí.

Původně byla v tabulce obsažena pole desetinných čísel velikosti 3. Pro větší přehlednost byla zavedena výše zmíněná třída. Tabulka byla vypočtena simulací mnoha soubojů. Dále třída obsahuje pomocné metody, které využívá AI. Například algoritmus, který zjistí, kolik je do daného území třeba přidat jednotek, aby pravděpodobnost dobytí nějakého území dosáhla zadané velikosti.

3.2.7 Propojení hlavních tříd

Nyní se podíváme na komunikaci mezi hlavními bloky aplikace. Konkrétně jde o GUI, AI a hlavní herní třídu Game. Obrázek 3.2 ukazuje zjednodušený model.



Obrázek 3.2: Komunikace hlavních tříd

Modře je znázorněno ovládání uživatelem. Každá jeho akce související se hrou vytvoří instanci třídy implementující rozhraní *IRiskAction*. Tento tah je ve třídě *Game* proveden díky metodě *applyAction*. Akce umělého hráče jsou v diagramu vyvedeny červeně. Nejprve GUI vznese požadavek na tah AI hráče. Třída AI jej vrátí ve formě instance *IRiskAction*. Tento tah se poté provede stejně jako v případě lidského hráče. Nakonec se GUI překreslí a vypíše akci, kterou hráč provedl.

4. Umělá inteligence

V této kapitole probereme návrh a realizaci umělých hráčů a opatření, která bylo nutno učinit, aby prohledávací strom nebyl moc široký a evaluační funkce dobře fungovala. Uvedeme si příklady herních situací, které AI neřeší ideálně. Dále si vypíšeme výsledky AI proti uživatelům, kteří hru testovali. Výsledky nakonec shrneme.

4.1 Umělá inteligence s prohledáváním

Většina umělých inteligencí do deskových her tohoto typu je implementována pomocí rozhodovacích pravidel. Tato práce proto od začátku tíhla k vyzkoušení hráče s prohledáváním.

4.1.1 Výběr hlavního města

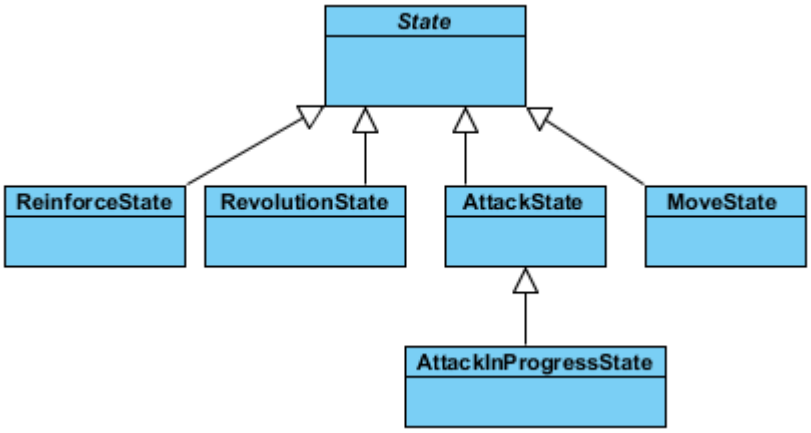
Metody pro výběr hlavního města stojí mimo prohledávání, poněvadž k němu dochází pouze jednou, a to na začátku hry. Třída AI obsahuje několik metod, jež z daných možností vyberou to hlavní město, které nejlépe umožní začít s plněním tajného úkolu. V průběhu testování hry se však ukázalo, že je výhodnější dobýt na začátku co nejvíce území místo plnění úkolu. Proto se na začátku hry volá taková metoda, která vybere území, jež je obklopeno největším počtem snadno dobyvatelných sousedů. Algoritmus předpokládá, že každé území je zpočátku osídleno 3, 5 nebo 7 neutrálními jednotkami. Každého kandidáta na hlavní město ohodnotí následovně: Projde všechny jeho sousedy a sečte jejich koeficienty podle funkce *coefficient*, která jako argument dostane počet jednotek. Funkce je uvedena níže.

$$\text{coefficient(x)} = \begin{cases} 3, & x = 3 \\ 1, & x = 5 \\ -1, & x = 7 \end{cases}$$

Algoritmus si vybere takového kandidáta, který bude mít nastřádanou hodnotu nejvyšší. Definice funkce zaručuje, že vybere takové území, které má slabě osídlené sousedy, a pokud takové území v nabídce není, vybere alespoň takové, jež má nejmenší počet silných sousedů. To může být užitečné, protože hlavní město nebude možné ohrozit z mnoha stran.

4.1.2 Stavy

AI pracuje s prohledávacím stromem, jenž se skládá ze stavů odpovídajících fázím hry. Proto vznikla abstraktní třída *State*. Na diagramu 4.1 vidíme hierarchii třídy *State* a jejích potomků.



Obrázek 4.1: Třída State a potomci

V následujících odstavcích si popíšeme důležité atributy a metody stavů. Bude popsán i způsob generování tahů v jednotlivých fázích.

State

Atributy a objekty

- `boolean hypotheticalValue` — takto označený stav už má ve stromě příliš velkou hloubku, nelze jej však ještě přímo ohodnotit.
- `State maxChild` — používá se k uchovávání potomka s nejlepší hodnotou evaluační funkce.
- `int cutoff` — tento atribut slouží k rozlišení koncových stavů. V případě nekoncového stavu má hodnotu -1, jinak nese ID hráče, který v daném stavu zvítězí. Strom se z koncového stavu již dále nerozvíjí.
- `int depth` — značí hloubku uzlu v prohledávacím stromu.
- `EvalStruct evaluation` — účelem pomocné třídy *EvalStruct* je ohodnocení stavu. Protože jde o hru více hráčů s nenulovým součtem, je hodnota stavu reprezentována polem desetinných čísel. Každý stav má odkaz na své ohodnocení.
- `boolean tainted` — tímto příznakem je označen takový uzel, který je ekvivalentní nějakému jinému uzlu ve stromě.
- `IRiskAction action` — je akce vedoucí do daného stavu. V případě, že stav je kořenem stromu, má hodnotu *null*.
- `ArrayList<State> children` — kolekce potomků daného stavu

Metody

- `double evalPlayer()` — vrátí hodnotu evaluační funkce pro daného hráče. Podrobně si ji rozebereme v sekci 4.1.4.
- `double [] evaluate()` — vrátí pole ohodnocení stavu. Počítá se pouze v listech. V ostatních uzlech se odvodí z potomků. K tomu, aby hráč škodil ostatním hráčům je třeba, aby hodnota stavu jednoho hráče brala v potaz i hodnoty ostatních hráčů. Přesný vzorec následuje níže.

$$\text{value(i)} = \text{playerValue(i)} - \frac{1}{act}\sum_{j=1}^{n}\text{playerValue(j)}$$

  kde *value(i)* je hodnota stavu pro i-tého hráče, *playerValue(i)* hodnota evaluační funkce i-tého hráče vrácená funkcí *evalPlayer()*, n počet hráčů na začátku hry, *act* počet hráčů v kořeni prohledávacího stromu. To je velice důležité, pokud bychom totiž vzali aktuální počet hráčů v daném uzlu, výpočet by se značně zkreslil. Může-li hráč zničit jiného hráče, bylo by pro něj výhodnější nechat jej hrát s hodnotou blízkou nule, tím by se od jeho vlastní hodnoty odečítala hodnota menší (protože by byla dělena mezi větší počet hráčů) a jeho výsledná hodnota by tím pádem byla větší.
- `double taskCompletion(Task t, Player p)` — spočítá, jak dobře zatím hráč splnil svou úlohu. Vrátí číslo mezi 0 a 1.
- `getChildren(HashMap<Integer, State> map)` — tato abstraktní metoda vytvoří a vrátí možné potomky stavu. Mapa v parametrech slouží k detekci již přítomných stavů ve stromě. V případě, že potomci již byli vytvořeni dříve, metoda je pouze vrátí.
- `State applyAction(IRiskAction action)` — důležitá abstraktní metoda, která stav změní pomocí dané akce, a tím vytvoří jiný stav (typicky v jiné fázi).
- `State createChild()` — pomocí metody *applyAction* vytvoří potomka a přidá jej do stromu. V základní variantě metoda již přítomné stavy vyřazovala pomocí asociativní mapy. Úskalí tohoto přístupu, včetně alternativního řešení, uvedeme v sekci 4.1.3.
- `int hashCode()` — spočítá numerickou hodnotu stavu, která poslouží jako klíč do asociativní mapy.

Nyní u každého potomka třídy State popíšeme, čím se liší a jak generují své tahy.

ReinforceState Tento stav slouží k přiřazování jednotek do území. Nechť s značí počet vojáků a t počet území. Vojáci jsou nerozlišitelní, na rozdíl od území, a proto celkový počet možností, jak přiřadit s vojáků do t území odpovídá vzorci:

$$\binom{t+s-1}{s-1}$$

Je vidět, že už v brzkých fázích hry je toto číslo příliš velké na to, aby bylo možné zkoumat všechny možnosti. Algoritmem vygenerujeme jen pár možných tahů.

Možné akce V tomto stavu je možné pouze přidat jednotky (nebo fázi přeskočit, ale to postrádá logiku). Přípustná akce může vypadat kupříkladu takto:

- ```
  AddSoldierPair pair1 = new AddSoldierPair(5,2);
  AddSoldierPair pair2 = new AddSoldierPair(10,8);
  new RiskActionAddSoldiers(new AddSoldierPair[]{pair1,pair2});
  ```

  Tato akce by přidala 2 jednotky do území s indexem 5 a 8 jednotek do území s indexem 10. Potomkem stavů tohoto typu mohou být buď stavy typu revoluce (pokud je ve stromě na tahu stejný hráč jako v kořeni), nebo útok (neznáme a nemůžeme znát karty jiného hráče, proto fázi přeskakujeme).

Generování tahů Algoritmus se nejdříve podívá na své revoluční karty a dopředu počítá s posilami, které mu případně přinesou. Ke svým výpočtům potřebuje asociativní mapu, jejíž klíče jsou indexy teritorií, která chceme doplnit vojáky, a hodnoty představují počet vojáků, jež má zatím území přisouzeno k přidání. Dále využívá haldu naplněnou instancemi třídy *ReinforceStruct*. Halda byla zvolena z toho důvodu, že vždy vybereme několik nejlepších řešení.

ReinforceStruct je pomocná třída, jež obsahuje údaje o přiřazování vojáků. Skládá se z indexů útočícího a napadeného území a počtu vojáků, kolik je třeba přidat k danému státu, aby byla decentní šance úspěšného dobytí/obrany. Je implementováno rozhraní *Comparable*, aby instance této třídy mohly být vkládány do haldy. Jako menší se bere taková instance, která má menší nároky na jednotky.

Popis jednotlivých přiřazení:

1. Snaha o dobytí co největšího počtu území. Není rozlišováno mezi neutrálními a nepřátelskými vojáky.

2. Přidání jednotek do území, která sousedí s nepřítelem. Pokus o prevenci nepřátelského útoku dostatečným zvýšením vojenské síly.

3. Přidání všech jednotek do hlavního města.

4. Přidání všech jednotek do jednoho území s nejvíce nepřátelskými sousedy.

Z algoritmického hlediska jsou zajímavé pouze prvé dvě možnosti. Obě si popíšeme blíže.

1. Algoritmus projde všechna vlastněná území a u každého se pomocí třídy *Probabilities* podívá, kolik by bylo třeba přidat jednotek, aby měl šanci dobýt souseda. To se vždy opakuje pro všechny sousedy. Pokud jsou nároky na jednotky menší nebo rovny počtu vojáků k dispozici, vytvoří se záznam *ReinforceStruct* a vloží se na haldu. Jsou přidávány i záznamy, které požadují 0 jednotek. Je to jednak proto, že přidáním jednotek zvýšíme pravděpodobnost vítězné bitvy a jednak také proto, že pokud by z jednoho území mohl hráč útočit na dvě slabá území, mohl být schopen dobýt jen jedno. Tuto situaci ilustruje obrázek 4.2.
  ```

Obrázek 4.2: Lze dobýt pouze jedno území

Zelený hráč má velikou šanci dobýt z Brazílie buď Argentinu nebo Peru, ale ne obě dvě země. Po úspěšném útoku by se totiž útočící jednotky přesunuly do dobytého území a v Brazílii by tak s velkou pravděpodobností zůstala pouze jediná jednotka. Po projití všech vlastněných teritorií začneme odebírat záznamy z haldy. Nejprve se přesvědčíme, zda jsme už území nedobyli odjinud. Pokud ano, vezmeme další záznam. Pokud již nezbývá dost jednotek ke splnění požadavku záznamu, smyčku přerušíme. V opačném případě si zapamatujeme jednotky, které se chystáme přidat, a budeme simulovat útok. Pomocí třídy *Probabilities* odhadneme, kolik nám po útoku zbude bojeschopných vojáků, a hodnotu si uchováme pro případ, že bude plánován další útok ze stejného území. Pokud však budeme chystat útok ze stejného území, je nutné znovu odhadnout počet potřebných jednotek, protože se změnil. Proces opakujeme, dokud není halda prázdná nebo nevyčerpáme dostupné jednotky. Nyní máme v asociativní mapě uloženy záznamy o tom, kolik kam přidat vojáků. Může se stát, že nějací zůstali nepřiřazeni. Zbylé vojáky rovnoměrně rozdělíme mezi kandidáty. Pak už jen zbývá vytvořit akci z dostupných údajů.

2. Algoritmus je obdobný s předchozím. Kromě již uvedeného se liší pouze tím, že odhaduje, kolik je třeba přidat k úspěšné obraně, a bere záznamy s nenulovým požadavkem na počet vojáků.

RevolutionState V tomto stavu lze hrát revoluční karty.

Možné akce

- `new RiskActionAddRev(card);`

  Do území s indexem *card* přidá revoluční jednotky.

- `new RiskActionBeginRev(card);`

  Na území s indexem *card* vyvolá revoluční útok.

- `new RiskActionRollAI();`

  Při rozpoutaném revolučním útoku hodí kostkami a odhadne, jak celý útok dopadne z pravděpodobnostní tabulky.

- `new RiskActionSkipPhase();`

  Přeskočí fázi a vytvoří tak stav reprezentující útočnou fázi hry.

- `new RiskActionSuperRevolution();`

  Zahraje všechny karty revoluce najednou.

Poznámka: Dříve byly akce vyvolání revoluce a hodu kostkou odděleny. Protože však při probíhající revoluci lze pouze útočit, obě akce se sloučily v jednu.

Generování tahů Pokud nemá hráč více karet, než je povolený limit, uvažuje pouze o zahrání karet, kde má alespoň 40% šanci na dobytí, v případě útoku na hlavní město stačí i menší šance. Přidávání revolučních jednotek se uvažuje vždy. Pokud by hráči mělo na konci kola zůstat karet více, než dovolují pravidla, hraje karty bez ohledu na pravděpodobnost a fázi nemůže přeskočit, dokud jeho počet karet neklesne na požadovanou úroveň. Super revoluce slouží ke zmenšení prohledávaného prostoru, ještě se k ní vrátíme.

AttackState Zde zvažujeme kandidáty k útoku.

Atributy a objekty

- `double p` — pravděpodobnost, že hod kostkou povede do tohoto stavu. Má smysl pouze v případě, že rodič je typu *AttackInProgressState*.

- `Heap<AttackStruct> attackCandidates` — halda, jež obsahuje záznamy o kandidátech na útok. Vybereme z množiny postupně několik nejlepších, proto byla zvolena halda.

- `ArrayList<AttackStruct> bestCandidates` — zapamatujeme si několik nejlepších kandidátů.

Možné akce

- ```
  AttackInfo atkInfo = new AttackInfo(atk, def, ind);
  new RiskActionBeginAttack(atkInfo);
  ```

  Napadne území s indexem *def* z území s indexem *atk*. Parametr *ind* značí pořadí v potomcích daného stavu.

- `new RiskActionSkipPhase();`

  Přesune se do fáze pohybu.
  ```

Generování tahů Nejdříve jsou vypočteni kandidáti k útoku, pokud nebyli předáni z minulých stavů. Algoritmus projde všechna vlastněná území a jejich sousedy. Pro každou dvojici, která splňuje podmínky (dostatečná šance na dobytí), je vytvořena instance třídy *AttackStruct* a vložena na haldu. Třída *AttackStruct* obsahuje index útočícího a napadeného území a pravděpodobnost dobytí. Pro rozlišení preferovaných útoků tato třída implementuje rozhraní *Comparable*. Porovnávaná kritéria od nejdůležitějšího:

1. Cíl je hlavní město, které ovládá původní vlastník.
2. Cíl není neutrální.
3. Pravděpodobnost dobytí

Následně z haldy vybereme předem určený počet nejlepších záznamů a vytvoříme potomky. U každého záznamu je třeba testovat, jestli se nezměnila pravděpodobnost (vrátíme do haldy s novou pravděpodobností) nebo jestli lze stále dobýt (cíl mohl být dobyt dřív, v tom případě záznam zahodíme).

AttackInProgressState Tato třída reprezentuje stav hry při probíhajícím útoku.

Atributy

- `int attackerIndex` — index útočícího území
- `int defenderIndex` — index napadeného území
- `int childNumber` — pořadové číslo uzlu v kolekci dětí svého rodiče

Dále dědí od *AttackState* například kandidáty k útoku.

Možné akce Protože možností, jak může dopadnout kombinace několika hodů kostkami, je mnoho, v zájmu redukce prohledávacího prostoru zkoumáme pouze dvě možnosti v případě hodu: dobytí/vyčerpání útočících jednotek.

- `new RiskActionRollAI(true);`

  Hodí kostkami a získá napadené území. Počet ztracených jednotek se odhadne podle tabulky z *Probabilities*.

- `new RiskActionRollAI(false);`

  Hodí kostkami a vyplýtvá jednotky bez dobytí. Obránci i útočníkovi se upraví počet přeživších jednotek podle tabulky z *Probabilities*.

- `new RiskActionRetreat();`

  Ihned přeruší útok.

Počítání hodnoty stavu s náhodou z hodnot jeho potomků je složitější než u ostatních stavů. V sekci 4.1.3 si o takových uzlech povíme víc.

Generování tahů Každá instance této třídy má maximálně 3 potomky. Všechny jsou druhu *AttackState*. Po jejich vytvoření vložíme do jejich haldy kandidátů všechny nepoužité nejlepší kandidáty z předchozího útočného stavu horší, než byl kandidát vedoucí do tohoto stavu. Jinak vyjádřeno, všechny stavy z *bestCandidates* v intervalu

$$(childNumber, |bestCandidates|).$$

MoveState V těchto stavech se rozhoduje, jaké území posílit přesunem jednotek.

Pomocné třídy

- `GraphComponenter` — rozdělí graf území na komponenty souvislosti. V jedné komponentě budou sousední území aktuálně hrajícího hráče.
- `GraphComponent` — reprezentuje jednu komponentu grafu souvislosti.
- `MoveDesirabilityStruct` — data o příjemci posílení. Obsahuje: index teritoria, které potřebuje jednotky, kolik jich potřebuje minimálně a kolik optimálně. Důležitost přesunu do daného území je dána atributem *desirability*. Jako podobné třídy, které jsme viděli, i tato implementuje rozhraní *Comparable*. Porovnávání probíhá výhradně na základě atributu *desirability*.
- `MoveDonorStruct` — tato třída slouží k reprezentaci potenciálních dárců jednotek. Skládá se pouze z indexu území, které může postrádat jednotky, a počtu dostupných jednotek. Opět je implementováno rozhraní *Comparable* a třídy se porovnávají přirozeně podle počtu jednotek, jež mohou postrádat.

Možné akce I přesunů může teoreticky existovat velké množství, omezíme se proto na několik málo, které algoritmus vygeneruje a bude považovat za nejlepší.

- ```
  MoveActionInfo move = new MoveActionInfo(from, to, toMove);
  new RiskActionMove(move);
  ```

  Přesune jednotky mezi dvěma svými sousedními územími. Čísla *from, to, toMove* značí indexy dárce, příjemce a počet jednotek, které se přesunou.

- ```
  new RiskActionSkipPhase();
  ```

  Přesune se do fáze posilování, na řadě bude další hráč.

Generování tahů

1. Algoritmus nejdříve rozloží graf území na komponenty s pomocí třídy *GraphComponenter*. Postupně prochází všechny komponenty a všechna jejich území. Pro každé území projde sousedy. U každého svého území vytvoří záznam *MoveDesirabilityStruct* pro možný útok i možné odražení nepřátelského útoku. Využije se k tomu pomocné třídy *Probabilities*. Minimální počet požadovaných jednotek z tohoto záznamu se užije pro obranu. Optimální počet zvýšený o minimální zase pro útok. Atribut *desirability* se určí podle minimálního potřebného počtu vojáků. Pokud hráč může

dobýt hlavní město, *desirability* se o mnoho zvýší. Takové území, které optimálně potřebuje alespoň jednu jednotku, se stane kandidátem na příjemce a jeho záznam se vloží do haldy příjemců. Dále se u území počítá, jestli mají přebytečné vojáky (například podle ohroženosti). Pokud ano, záznam o něm se vloží do haldy dárců. Poté, co naplníme obě haldy, zkusíme plnit požadavky záznamů z haldy příjemců. Pro každý požadavek prohledáme kopii haldy dárců pro nejvhodnějšího a nejbližšího dárce. Halda dárců musí být zkopírovaná, protože ji pro každého příjemce můžeme celou vyprázdnit. Opakujeme, dokud není halda požadavků prázdná nebo jsme nenalezli 3 možné tahy.

2. Posílení hlavního města — pokud je hlavní město ohroženo, pokusí se do něj přesunout nějaké jednotky.

4.1.3 Vlastnosti stromu hry

V této sekci jsou analyzovány vlastnosti prohledávacího stromu a problémy, které bylo nutno překonat. Dále jsou uvedena řešení vedoucí ke zmenšení prohledávaného prostoru.

Zmenšení stromu Prohledávaný prostor může nabývat neúnosných rozměrů, proto byly provedeny pokusy o jeho redukci.

1. Dopředné prořezání. Jsou uvažovány pouze některé tahy. Toto téma již bylo popsáno v minulé sekci, v generování tahů.

2. Ekvivalence stavů. U každého přidávaného uzlu se pomocí asociativní mapy testuje, zda už není ve stromě přítomen. Rozlišujeme následující případy:

   (a) Uzel není přítomen → přidáme.

   (b) Na cestě mezi uzlem a kořenem se vyskytuje ekvivalentní uzel → zahodíme. Odpovídá situaci, kdy hráč zaútočí a ihned uteče.

   (c) Ekvivalentní uzel má stejného rodiče → zahodíme. Například, když se vícekrát aplikuje na jeden stav stejná akce.

   (d) Ekvivalentní uzel už je ohodnocen → nový uzel přidáme a nastavíme příznak *tainted*. Ohodnotíme podle jeho ekvivalentu.

   (e) Ekvivalentní uzel ještě není ohodnocen → prohlásíme jej za *tainted*, nový uzel vložíme místo něj do asociativní mapy a zkopírujeme referenci na ohodnocení. Tím zajistíme, že v okamžiku ohodnocení budou ohodnoceny oba stavy.

3. Symetrie. Pořadí útoků se dá v jistých případech pokládat za ekvivalentní. Zřejmé případy, kdy útoky nemají společného ani útočníka ani obránce, zachytí asociativní mapa. Ostatní případy, kdy se útočí z jednoho území, už samozřejmé nejsou. Například, chce-li hráč útočit z území A do B a následně do C, lze předpokládat, že pokud je v A dostatečný počet jednotek, platí:

$$(A \to B, A \to C) \approx (A \to C, A \to B)$$

Avšak po aplikaci stejných akcí v různém pořadí AI vytvoří stavy, které se v daném území liší byť o jednu jednotku. Je to způsobeno zkresleným odhadem z pravděpodobnostních tabulek. Symetrie útoků jsou řešeny pomocí nerostoucí posloupnosti útočných kandidátů. To znamená, že pokud se v jedné větvi provede útok na jednoho kandidáta, v té samé větvi ani ve větvích od ní napravo v tom samém podstromu už nemůže být proveden útok na kandidáta, jenž je považován za lepšího. Technické provedení bylo uvedeno v *Generování tahů u AttackInProgressState*.

4. Super revoluce. Ve fázi posilování je důležité, aby hráč viděl minimálně do útočné fáze a mohl tak naplánovat útoky. Pokud však má mnoho karet revoluce, nikdy se tak hluboko nedostane. Proto ve stromě v revoluční fázi zahraje jednou akcí všechny své karty a odhadne výsledek.

Algoritmus Minimax Tento algoritmus se používá pro strategické hry dvou a více hráčů. Jeho účelem je nalezení nejlepšího tahu v dané pozici. Algoritmus pro správné fungování potřebuje funkci na ohodnocování stavů. Vygeneruje strom tahů do určité hloubky a listy ohodnotí pomocí evaluační funkce. Dále se postupuje po vrstvách od listů až ke kořeni stromu. Hodnota každého uzlu se určí jako maximum z jeho potomků tak, aby maximalizovala hodnotu hráče, který je v daném uzlu na tahu. Tímto se nejlepší hodnota pro hráče na tahu dostane až do kořene stromu a hráč již pouze vybere akci vedoucí do potomka, jenž má stejnou hodnotu jako kořen. Často se eliminují podstromy, kam nevede optimální strategie, jde o tzv. alfa-beta prořezávání (viz přednáška 7 [5]).

Problémy minimaxu pro hru Risk Risk se však od mnoha deskových her liší například tím, že každý hráč odehraje najednou několik akcí. Hráči se tedy nestřídají po každé provedené akci. Situací, kdy se v prohledávacím stromě změní hráč, je tím pádem velice málo. Z toho plyne, že například alfa-beta prořezávání by nepřineslo téměř žádné zmenšení. Další odlišností je výskyt náhody v prohledávacím stromě. Hlavní problém však tkví v tom, že ve stejné hloubce mohou být uzly, které představují různé fáze tahu.

Příklad Pokud jedna větev stromu dojde až do posilovací fáze dalšího hráče a jiná větev nikoliv, zkreslí se značně ohodnocení stromu. Onen hráč získá bonus za nové jednotky, a protože hodnota stavu se počítá podle všech hráčů (viz vzorec v sekci *Stavy*), sníží hodnotu stávajícího hráče, a tím ovlivní jeho rozhodnutí. Situacím, kdy se náhle změní ohodnocení, se říká *efekt horizontu*.

Řešení efektu horizontu V projektu je problém řešen tak, že všechny listy musí být ve stejné fázi a na tahu musí být stejný hráč. To však není jednoduché zaručit. Například koncové stavy nebo stavy s příznakem *tainted* už se dále nerozvíjí, ať jsou v jakékoliv fázi.

- Koncový stav se řeší bonusem pro vítězného hráče. Tím se překoná potenciální zkreslení ohodnocení v jiné než cílové fázi.
- Tainted stav — z odstavce *Zmenšení stromu* víme, že tyto stavy budou ohodnoceny správně podle svých ekvivalentů.

Třída *PhaseCounter* určí, v jaké fázi má strom ukončit prohledávání. Podle parametru *DEPTH_SKIPS* ve třídě *AI* bude prohledávat ne do hloubky uzlů, ale do hloubky fáze. Může se stát, že se strom stane příliš hlubokým, aniž by se dostal do cílové fáze. Stává se to hlavně v pozdějších stádiích hry, kdy má hráč mnoho dobrých možností kam útočit. V případě, že strom překročí mezní hloubku definovanou ve třídě AI, se uzel ohodnotí pomocí singulárního prodloužení: Prohledávání probíhá už jen do hloubky 1. Pro každý uzel se vygenerují potomci a ihned se ohodnotí. Vybere se ten nejlépe ohodnocen, který rozvíjíme dále stejným způsobem. Proces se opakuje, dokud větev nedosáhne cílové fáze. Stavy v útočné fázi za mezní hloubkou se rovnou přeskakují. Jakmile prohledávání dosáhne cílové fáze, ohodnotí list a toto ohodnocení přiřadí uzlu před mezní hloubkou. Takto ohodnocenému uzlu se nastaví příznak *hypotheticalValue*, protože hodnota byla spočtena z uzlu, který se ve stromě nevyskytuje.

Uzly náhody V prohledávacím stromu jsou i uzly ovlivněné náhodou. Konkrétně jde o uzly, kde hráč hází kostkou. Hráč se rozhoduje, zda se vyplatí riskovat ztrátu jednotek vykoupenou případným ziskem území.

Ohodnocení Hodnota stavů s náhodou se počítá dle vztahu:

$$\max\{r_1 * p + r_2 * (1 - p), ret\}$$

kde r_1 značí hodnotu z podstromu po vítězném hodu, p pravděpodobnost výhry, r_2 hodnotu po prohraném hodu, ret hodnotu z podstromu v případě zanechání útoku. Očekávaný zisk nemusí být nejlepším pomocníkem při volbě tahu, jak ukazuje následující odstavec.

Užitek Informace v tomto odstavci jsou čerpány z přednášky o umělé inteligenci[6](přednáška 6).

- Funkce užitku mapuje stavy/loterie na reálná čísla.
- Proč nejsou peníze přímou mírou užitku? Budeme uvažovat, že vyhrajeme 1 mil. USD. a můžeme si ho buď nechat, nebo přijmout sázku na hod mincí: padne-li orel dostaneme 2,5 mil. USD, jinak nic. Jak se rozhodnout?
- Očekávaný zisk při sázce je 1.250.000 USD, většina lidí však volí jistotu 1.000.000 USD. Je to racionální?
- Volba v předchozí hře závisí nejen na hře, ale i na současném majetku hráče.
- Nechť S_n je stav označující majetek n USD. Potom očekávaný užitek akcí můžeme napsat jako:

$$\begin{aligned}\text{EU(ACCEPT)} &= \frac{1}{2}U(S_k) + \frac{1}{2}U(S_{k+2.500.000})\\ \text{EU(DECLINE)} &= U(S_{k+1.000.000})\end{aligned}$$

- Nechť $U(S_k) = 5$, $U(S_{k+1.000.000}) = 8$, $U(S_{k+2.500.000}) = 9$. Potom je rozhodnutí odmítnout sázku zcela racionální.

Alternativy V průběhu tvorby projektu byly uvažovány i jiné možnosti rozhodování. Například pomocí mezní hodnoty: Napevno by se určila minimální hodnota, o kterou se musí zvýšit hodnota stavu, aby se hráč rozhodl házet. Pokud by zisk násobený pravděpodobností dobytí překročil mezní hodnotu, hráč by házel. Zisk by byl spočten jako rozdíl hodnoty stavu po dobytí (spočítané rekurzivně) a hodnoty (přímo spočtené) před hodem. Určit mezní hodnotu by bylo obtížné, navíc by se v průběhu hry měnila. Proto zůstalo u původní varianty.

Škálování funkce Na začátku hry může funkce růst exponenciálně s časem. Čím více má hráč území, tím více dostává jednotek v každém tahu. Skutečnému užitku lépe odpovídá funkce rostoucí lineárně v čase, a proto hodnotu funkce zlogaritmujeme.

4.1.4 Evaluační funkce

Ke spočtení hodnoty stavu je nejdříve třeba ohodnotit každého hráče. K tomu slouží evaluační funkce. Tato funkce prošla určitým vývojem, který si v této části práce přiblížíme.

Základní varianta V prvních variantách brala funkce v potaz hlavně:

1. materiální hledisko: počet starých a nových karet, bonusy za vlastněné kontinenty, počet vojáků. Velký význam byl přikládán počtu vlastněných území.
2. úkol: Procentuálně vyjádřené splnění úkolu je vynásobeno konstantou a přičteno k průběžné hodnotě.
3. bonus za výhru

Vylepšení Později byl přidán i poziční aspekt. Nedá se očekávat, že se ve stromě dostaneme dostatečně hluboko, aby se ohrožení vlastněných území proměnilo v jejich ztrátu. Zvláště pokud je ohroženo hráčem, který bude na tahu jako poslední. Následující odstavce popisují všechna vylepšení od základní varianty.

Poziční hledisko

1. Hráč projde všechna svá území a pro každé sousední, které nevlastní, přičte k hodnotě funkce převrácený počet vojáků. Tím se jednak odliší dříve stejně ohodnocené stavy, jednak budou lépe ohodnoceny stavy, které dávají lepší šanci na dobytí území.
2. Hráč prochází všechna svá území a jejich sousedy. Pokud je pravděpodobnost dobytí větší než 0.5, vynásobí se kladnou konstantou a přičte k dosavadní hodnotě. Naopak, pokud je území ohroženo, součin pravděpodobnosti s konstantou se odečte. V případě hlavních měst jsou konstanty větší.

Karty revoluce V počátcích funkce počítala s bonusy za staré i nové karty revoluce. Hlavně na začátku hry je ale lepší hrát všechny karty, proto byl odstraněn bonus za staré karty. Tím je hráč veden k brzkému používání karet.

Hlavní město Hlavní město má stěžejní úlohu, proto se mu věnujeme i v evaluační funkci. Nejprve spočítáme koeficient ohroženosti:

$$koeficient = \begin{cases} 3, & \text{existuje nepřátelský soused} \\ 1, & \text{existuje nepřátelské území se vzdáleností 2} \\ 0, & \text{ostatní případy} \end{cases}$$

1. Bonus dle koeficientu: $bonus = (3 - koeficient) * 150$. Koeficient bere v potaz i možné ohrožení z jiných než bezprostředně sousedních území. Díky tomu se AI hráč bude snažit zbavit všech nepřátel v okolí svého hlavního města.

2. Bonus za jednotky: Aby hráč náhle neprohrál kvůli překvapivému útoku revoluce, dostává bonus za jednotky na území hlavního města. To však může vést k přibrzdění rozvoje. Proto se rozlišují dvě situace.

   (a) Hráč má nejvíce území: Bonus dostane za každou jednotku, maximálně však za 10. Pokud hráč vyhrává, může si dovolit obětovat nějaké jednotky na obranu. Ztráta hlavního města v pozdějších fázích hry bývá fatální. Bonus je shora omezen, protože jinak by hráč vždy považoval za lepší posílit hlavní město, než někam zaútočit.

   (b) Jinak bonus dostane maximálně za 3 jednotky. I když hráč nevyhrává, je dobré myslet i na obranu.

   Jednotky v hlavním městě je třeba řešit už v posilovací fázi. Přidání malého počtu jednotek se dá provést třeba tak, že se do množiny území, mezi které se rozdělí zbylé jednotky, přidá i hlavní město.

3. Penalizace za znovudobytí: Dobývat zpět své ztracené hlavní město nemá žádné výhody, právě naopak. Proto je hráč citelně penalizován za pokus o znovuzískání hlavního města. Funkce od penalizace upustí pouze v případě, že jde o koncový stav, nebo pokud svou silou výrazně převyšuje všechna okolní nepřátelská území .

4.1.5 Problémové herní situace

Zde budou popsány situace, které AI neřeší dobře. Některé z nich se podařilo ošetřit.

- V některých situacích, kdy dobývá území a šance na dobytí se blíží jedné, se rozhodne útok přerušit a zaútočí jinam. Počítá totiž s tím, že je bude možné dobýt i potom. Pokud se AI rozhodne přerušit útok a potom opět útočit ze stejného území, přerušení útoku potlačíme.

- Výpočet v uzlech náhody by mohl být hodně zkreslený, pokud by z důvodu ekvivalence stavů chyběl nějaký potomek. To napravuje zavedení příznaku *tainted*.

- Ve snaze chránit hlavní město si hráč přesune jednotky do hlavního města z území, které je samo ohroženo. Zde nutně nemusí jít o chybu, pouze pokud existuje lepší území, odkud podniknout přesun.

- Hráč dobyde cizí hlavní město, i když v příštím kole ztratí to své. Z hlediska revolučních karet by bylo moudřejší s útokem počkat.
- Pokud je šance na dobytí příliš malá, o útoku se vůbec neuvažuje. Hráč je tak připraven o možnost z jednoho území cíl oslabit a z dalšího následně dobýt. Ve hře tato situace téměř nenastává. Speciálně je ošetřen případ, kdy je ohroženo hlavní město. AI útočí všemi možnými prostředky, aby se ohrožení zbavila.

4.2 Přímočará umělá inteligence

Tato část je zasvěcena popisu jednoduchého přímočarého umělého hráče. Každý agent implementuje rozhraní *Lux*. V této práci byl však implementován pouze jeden. Rozhraní i jeho implementace se řídí zdrojovými kódy z dílny Sillysoft [7], až na situace plynoucí z odlišnosti návrhu a pravidel této implementace od jiných. Následuje přehled důležitých metod z rozhraní.

- `Territory chooseTerritory` — vybere si z nabídky území hlavní město.
- `IRiskAction revolutionPhase` — vrátí tah, který lze hrát v revoluční fázi.
- `IRiskAction placeArmies` — vrátí přiřazení jednotek v posilovací fázi.
- `IRiskAction attackPhase` — zahraje jeden tah v útočné fázi.
- `IRiskAction movePhase` — zahraje tah ve fázi pohybu.
- `boolean roll` — podle této metody se hráč rozhodne, zda bude házet kostkou nebo přestane útočit.
- `IRiskAction doMove` — provede jeden tah v aktuální fázi hry.

4.2.1 Výběr hlavního města

Preferováno je území na prázdném kontinentu. Pokud takové nelze vybrat, je vybráno území náležící nejmenšímu kontinentu.

4.2.2 Tahy

Zde bude popsána jednoduchá strategie, s jakou se hráč rozhoduje pro tahy v jednotlivých fázích kola.

Fáze posilování Agent nejdříve zkontroluje, jestli se v okolí jeho hlavního města nevyskytuje nepřátelské území až do vzdálenosti 2. Pokud ano, přidá si všechny dostupné jednotky do hlavního města. Jinak vyhledá území, které má nejvíce nepřátelských sousedů. V případě, že takové území neexistuje, zvolí zemi s nejvíce neutrálními sousedy. Do zvoleného území přiřadí všechny jednotky.

Fáze revoluce Hráč ihned použije všechny dostupné karty revoluce.

Fáze útoku Útoky se řídí prioritami. Zde je uvádíme od největší po nejmenší.

1. Útok na hlavní město
2. Útok na nepřítele
3. Útok na neutrální stát

Aby hráč zaútočil, musí mít území, odkud útok podniká, více jednotek než území napadené. Jakmile se jednou rozhodne k útoku, pokračuje v něm, dokud nezvítězí, nebo nepřijde o všechny jednotky.

Fáze pohybu V této fázi najde své území, které má nejvíce jednotek a zároveň nemá v okolí nepřátele. Tyto jednotky přesune do souseda, který je obklopen nejvíce nepřáteli. Pokud nelze takový tah uskutečnit, fázi přeskočí.

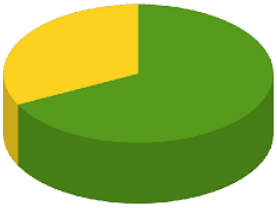
4.3 Experimentální výsledky

Oba typy umělé inteligence byly pouštěny proti sobě i proti lidským hráčům. Tato část odhaluje výsledky experimentů. Celkem hru testovalo 10 lidí. Byli mezi nimi jak úplní začátečníci, tak i zkušení hráči. Přehled výsledků nalezneme v tabulce.

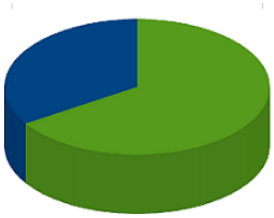
Hráči	**Skóre**
Main AI: Plain AI	527:253
Main AI: Člověk	56:29
Plain AI: Člověk	28:35

Tabulka 4.1: Výsledky experimentů

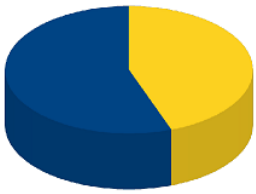
Obrázek 4.3 znázorňuje grafy vzniklé z dat tabulky.



(a) Main AI: Plain AI



(b) Main AI: Člověk



(c) Plain AI: Člověk

Obrázek 4.3: Grafické vyjádření výsledku her. *Main AI* má zelenou barvu, *Plain AI* žlutou a lidský hráč modrou.

4.4 Analýza výsledků

Z výše uvedeného je vidět, že AI s prohledáváním hraje o něco lépe než přímočará. Vedla si dobře i proti lidem. Výsledky dopadly nad očekávání. Šířka větvení a náhodné faktory jako nepředvídatelnost karet nebo hracích kostek spíše ukazuje na možnost, že prohledávání by nepřineslo nic navíc. Dalším problémem se zdála být potenciálně velká hloubka prohledávacího stromu způsobena faktem, že se hráči nestřídají po jedné akci, ale po jejich sérii. Výsledky však musíme brát s rezervou právě kvůli náhodnosti.

5. Závěr

5.1 Srovnání s podobnými projekty

Všechny testované implementace se drží původních pravidel deskové hry. Tato implementace Risku má podstatně změněná pravidla, což ji činí unikátní. Na druhou stranu znemožňují odlišnosti pravidel hlouběji srovnávat kvalitu AI hráčů u ostatních implementací. Porovnáme je alespoň z uživatelského pohledu.

5.1.1 Lux Delux

Tato komerční verze klasického Risku nabízí mnoho různých map a umělých hráčů. Pyšní se hezkou grafikou a intuitivním ovládáním. Vyzdvihl bych nápad na mapě zobrazovat počet jednotek, se kterými lze přímo manipulovat. Pokud je tedy v území pouze jedna jednotka, vůbec se nezobrazí, což má pozitivní vliv na přehlednost. Další příjemnou věcí je nenápadná tabulka, která ukazuje, kolik má kdo území, jednotek za kolo a další užitečné informace. Plusem je také možnost hraní po síti. Trochu horší už to bylo s umělou inteligencí. Ze škály hráčů spadajících do těžké obtížnosti byl důstojným soupeřem jen jeden, a to pouze do doby, než jsem se s tímto provedením hry sžil. Možná, že v plné verzi se vyskytují schopnější AI hráči. Dále mi zde chyběl nějaký panel zpráv, protože hra je rychlá a mnohdy si člověk nevšimne, co se děje.

Obrázek 5.1: Lux Delux

5.1.2 Domination

Další hra na motivy klasického Risku je volně šiřitelná. Také disponuje mnoha mapami a možností hrát přes internet. U této implementace je AI o mnoho lepší než u hry *Lux Delux*. Na zkušeného hráče však stále nestačí. Z několika odehraných her vyplynulo, že hráč, který začíná, má opravdu velkou výhodu a pro ostatní je velice těžké ji vykompenzovat. Risk se změněnými pravidly, kterému je věnována tato práce, zmíněnou vadou netrpí. Na druhou stranu je pro pokročilé hráče dobrou výzvou nechat hrát AI jako prvého.

Obrázek 5.2: Domination

5.2 Zhodnocení projektu

Hlavním cílem práce bylo převést deskovou hru na obrazovky počítačů a vyvinout pro ni umělou inteligenci založenou na prohledávání. I když není AI bez nedostatků, dokáže potrápit i protřelé hráče. Druhotným záměrem bylo implementovat přímočarého hráče, který se řídí pouze jednoduchými pravidly bez prohledávání, a následně oba hráče porovnat. Všechny tyto cíle se podařilo splnit. Začínající hráči, kteří neradi čtou návody, ocení začátečnický režim s přehlednou nápovědou. Hráč dostává nápovědu nejen na začátku každé fáze, ale i pokud se pokusí zahrát typické začátečnické chyby (například chybný výběr teritorií při útoku). Stinnou stránkou projektu je však návrh. Jak program narůstal, ukázalo se, že mohl být navržen o mnoho lépe. Dnes bych již spoustu věcí řešil jiným způsobem.

5.3 Budoucnost projektu

Program sice funguje dobře, nicméně se dá vymyslet nejedno vylepšení. Za vyjmenování stojí například:

- Paměť na karty — hráč by si pamatoval, které karty byly zahrány. Z toho by věděl, jaká území není třeba úzkostlivě chránit.
- Nevyvolávat revoluce u území, která pravděpodobně dobyde soupeř v příštích tazích.
- Větší zaměření na plnění úkolů
- Implementovat složitějšího přímočarého hráče.
- Upravit pravidla tak, aby bylo výhodné znovu dobýt hlavní město.
- Ošetření některých situací z 4.1.5
- Zatraktivnit grafiku, přidat nějaký efekt kupříkladu u házení kostek (animace změny čísel).

Seznam použité literatury

[1] Parker Brothers, Risk, The world conquest game
http://www.hasbro.com/common/instruct/risk.pdf

[2] Erik Arneson, History of Risk
http://boardgames.about.com/od/risk/a/risk_history.htm

[3] Hasbro games - Risk
http://www.hasbro.com/games/en_US/risk/

[4] Stuart Jonathan Russell, Peter Norvig, Artificial intelligence: A Modern Approach, Prentice Hall, 2003

[5] Roman Barták, přednáška Umělá inteligence I
http://kti.mff.cuni.cz/~bartak/ui/index.html

[6] Roman Barták, přednáška Umělá inteligence II
http://kti.mff.cuni.cz/~bartak/ui2/index.html

[7] Sillysoft SDK
http://sillysoft.net/sdk/

Seznam tabulek

Seznam použitých zkratek

AI - Artificial Intelligence, umělá inteligence
GUI - Graphic User Interface, uživatelské grafické rozhraní
ID - Identifikační číslo

Přílohy

Součástí projektu je kompaktní disk, který obsahuje:

- bc_risk.pdf — elektronická verze tohoto textu
- složka Distribuce Risk — aplikace připravená pro distribuci. Součástí je i spustitelný soubor Risk.jar.
- složka MyRisk — obsahuje celý projekt i se zdrojovými soubory.
- složka javadoc — vývojová dokumentace vytvořená stejnojmenným nástrojem